Universidade do Minho
Escola de Engenharia

Ângelo Semedo Nhaga

Desenvolvimento de um sistema de aquisição de sinal sem fios para aplicação no desporto

Dissertação de Mestrado Engenharia Eletrónica Industrial e de Computadores
Trabalho efetuado sob orientação de
Professor Hélder Carvalho
Professor Paulo Cardoso

Junho 2020

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Atribuição-NãoComercial-SemDerivações
CC BY-NC-ND

https://creativecommons.org/licenses/by-nc-nd/4.0

[Esta é a mais restritiva das nossas seis licenças principais, só permitindo que outros façam download dos seus trabalhos e os compartilhem desde que lhe sejam atribuídos a si os devidos créditos, mas sem que possam alterá-los de nenhuma forma ou utilizá-los para fins comerciais.]

Agradecimentos

Esta dissertação e todo o meu percurso académico foi feito de esforço e empenho. Mas a verdade é que algumas pessoas possuíram papeis preponderantes nas minhas conquistas.

A DEUS. Aos meus pais que sempre me apoiaram, que me ergueram quando caí, que nunca desistiram de mim, que me tornaram um verdadeiro "lutador" para toda a vida e, que fizeram de mim a pessoa que sou hoje. Que ORGULHO em vos ter como PAIS! Sem vocês nada seria possível!! Aos meus irmãos por estarem sempre comigo, espiritualmente, por terem-me apoiado, pelas palavras nos momentos de desespero e de maior dificuldade. Vocês são os melhores irmãos que podia pedir a DEUS!! À minha namorada por partilhar esta longa caminhada comigo, por não me deixar baixar a cabeça, em momento algum, por me motivar todos os dias, por ser incondicional, pela dedicação, o conforto e, pelo amor que me deu todos os dias. AMO-TE! À minha família, na sua generalidade, mas em especial à minha avó Ângelina, ao meu pai Jorge (as ESTRELAS que me guiam), aos meus tios Alexandre e Antónia, pelos valores e ideais que me incumbiram, por me darem um apoio incondicional em qualquer momento. Esta conquista também é vossa!

Aos meus orientadores por todo os esforços que empenharam no resultado da investigação, pelas sugestões, dicas e conselhos, pelo excelente trabalho de orientação e, pela exigência e
seriedade que demonstraram durante todo o percurso. Obrigada Universidade do Minho,
por todas as experiências, conhecimento e valores transmitidos! Finalmente, a todos os que, direta ou indiretamente, interagiram comigo e fizeram e fazem parte da minha vida e do meu percurso um OBRIGADO do tamanho do mundo!!!

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Assinado por : **ANGELO SEMEDO NHAGA**
Num. de Identificação: BI325019738
Data: 2020.07.03 11:36:42+01'00'

CARTÃO DE CIDADÃO

Resumo

Atualmente é cada vez mais comum a utilização de dispositivos tecnológicos no desporto que permitem auxiliar o desempenho desportivo dos seus utilizadores. Como por exemplo podemos referenciar o *Babolat Pure Drive Play Racket, miCoach Smart Ball Sony Smart Tennis Sensor, Swing Tracker, Lumo Run Running shorts and capris, Wilson X connected basketball,* entre outros[1]. Para além de auxiliarem a performance, estas tecnologias permitem também monitorizar a pressão arterial, frequência cardíaca, temperatura corporal e calorias queimadas. Estes tipos de tecnologias são utilizadas em diversos desportos, sendo as artes marciais exemplo da aplicação destas tecnologias e existindo já no mercado algumas aplicações, como são o exemplo do *Corner* [1], *Hykso Trackers*[2] e *Impactwrap*[3].

O sistema de aquisição desenvolvido tem capacidade de adquirir sinal e transmiti-lo em tempo real para um cliente, aplicação ou outro dispositivo via BLE (Bluetooth Low Energy). Este sistema pode ainda servir de *framework* para outras aplicações que envolvem a aquisição de sinal, como o ciclismo, o *baseball,* ou até o *rugby*.

Dos componentes a utilizar, na implementação do sistema, pode destacar-se o microcontrolador (CC2640R2F)[4] que já possui a funcionalidade de comunicação Bluetooth, o qual permite a obtenção do sinal e conversão analógico-digital dos dados que depois serão posteriormente enviados para um dispositivo externo (telemóvel, ou *tablet*) via BLE (Bluetooth Low Energy).

O projeto desenvolvido é um sistema de aquisição de sinal usando o BLE constituído por um serviço que possui duas características uma responsável pela receção de dados e a outra responsável pelo envio de dados, através do Bluetooth Low Energy (BLE), para um dispositivo externo, nomeadamente um dispositivo Android. Tendo em conta os exemplos acima referidos, o *Corner*[1] e o *Hykso Trackers*[2], algumas funções deste projeto foi pensada para a aplicação no boxe.

Palavras-chaves: *Real Time Operating System*, Desporto, Bluetooth Low Energy, Android, Sensor, Aquisição de Sinal.

Abstract

Nowadays it is more and more common to use technological devices in sport that help support the sporting performance of its users. As an example, we can refer to Babolat Pure Drive Play Racket, miCoach Smart Ball Sony Smart Tennis Sensor, Swing Tracker, Lumo Run Running shorts and capris, Wilson X connected basketball, among others [1]. In addition to assisting performance, these technologies also allow you to monitor blood pressure, heart rate, body temperature and calories burned. These types of technologies are used in several sports, with martial arts being an example of the application of these technologies and there are already some applications on the market, such as Corner [1], Hykso Trackers [2] and Impactwrap [3].

The developed acquisition system is capable of acquiring a signal and transmitting it in real time to a customer, application or other device via BLE (Bluetooth Low Energy). This system can also serve as a framework for other applications that involve signal acquisition, such as cycling, baseball, or even rugby.

Of the components to be used in the implementation of the system, the microcontroller (CC2640R2F) [4], which already has the Bluetooth communication functionality, which allows obtaining the signal and analog-digital conversion of the data, which will later be sent, can be highlighted. to an external device (mobile phone, or tablet) via BLE (Bluetooth Low Energy).

The developed project is a signal acquisition system using BLE consisting of a service that has two characteristics, one responsible for receiving data and the other responsible for sending data, via Bluetooth Low Energy (BLE), to an external device, namely an Android device. Taking into account the examples mentioned above, Corner [1] and Hykso Trackers [2], some functions of this project were designed for application in boxing.

Keywords: Real Time Operating System, Sport, Bluetooth Low Energy, Android, Sensor, Signal Acquisition.

Índice

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros ii

Agradecimentos iii

Declaração de Integridade iv

Resumo v

Abstract ix

Índice de Figuras xiii

Acrónimos xv

Capítulo 1 – Introdução 2

1.1 Motivação e Contextualização 2

1.2 Objetivos 3

1.3 Estrutura da Dissertação 4

Capítulo 2 – Estado da Arte 5

2.1 Trabalhos Relacionados 5

2.1.1 *Low Cost Wireless Data Acquisition System for Multi- sensor Applications* 5

2.1.2 *Wireless Data Acquisition System for IoT Applications* 6

2.1.3 Impact Wrap 7

2.1.4 Corner 7

2.1.5 Hykso Trackers 8

2.2 Bluetooth 8

2.2.1 Camada Lógica 10

2.3 Bluetooth Low Energy 11

2.3.1 Bluetooth 4.2 14

2.3.2 Camada Física 14

2.3.3 Camada Ligação 15

2.3.4 Host Controller Interface (HCI) 16
2.3.5 Logical Link Control and Adaptation Layer Protocol (L2CAP) 16
2.3.6 Maximum Transmission Unit (MTU) 18
2.3.7 Data Length Extension (DLE) 19
2.3.8 Camada de Gestão de Segurança 19
2.3.9 Attribute Protocol (ATT) 20
2.3.10 Generic Attribute Profile (GATT) 21
2.3.11 Generic Access Profile (GAP) 22
2.4 Sistemas Operativos em tempo real para Sistemas Embebidos: caso do TI-RTOS 25
2.4.1 TI-RTOS 26
2.4.2 Interrupções de hardware (Hwi) 27
2.4.3 Interrupções de software (Swi) 27
2.4.4 Tarefas 27
2.4.5 ICall 28
2.5 Microcontrolador CC2640R2F 30
Capítulo 3 – Análise do Problema 32
3.1 Definição do Problema 32
3.2 Requisitos 32
3.3 Caso de uso 33
3.4 Diagramas de Sequência 34
3.5 Arquitetura do Sistema 36
Capítulo 4 – Especificação e Implementação do Sistema 37
4.1 Especificação da Solução 37
4.1.1 Especificação da Solução 37
4.1.2 Receção e Processamento do Sinal 41

4.2 Implementação 42
4.2.1 Aquisição e Envio – Placa TI 42
4.2.2 Comandos 45
4.2.3 Trama 47
4.2.4 Receção e Processamento – Dispositivo Móvel 47
Capítulo 5 – Testes e Resultados 50
5.1 Resultados 50
5.1.1 Testes às taxas de transmissão 50
5.1.2 Testes ao Envio e Receção 50
5.1.3 Testes aos Comandos 51
5.1.4 Evolução da perda de dados nos canais de amostragem 52
5.1.5 Aplicação Android 53
Capítulo 6 – Conclusão 54
6.1 Limitações 54
6.2 Trabalhos Futuros 54
6.3 Considerações Finais 55
Referências 56

Índice de Figuras

Figura 1 - Diagrama de bloco do sistema[4] 6
Figura 2 Diagrama de bloco do sistema[5] 6
Figura 3 Impact Wrap 7
Figura 4 Corner 7
Figura 5 Hykso Trackers 8
Figura 6 Exemplos de topologias de piconet mestre/escravo[6] 8
Figura 7 Classes do Bluetooth[7] 9
Figura 8 Estrutura dos pacotes[8] 11
Figura 9 Funcionamento entre as cargas do protocolo GAP[9] 12
Figura 10 Troca de dados entre um dispositivo periférico e um central[9] 12
Figura 11 Organização da base de dados do Bluetooth BLE[9] 13
Figura 12 Distribuição dos canais na banda ISM[10] 14
Figura 13 Terminologia Geral do L2CAP (Adaptado [11]) 17
Figura 14 Arquitetura L2CAP (Adaptado [11]) 18
Figura 15 Pacote L2CAP mais Protocolo de Atributo (Adaptado[11]) 18
Figura 16 Tamanho e tempo do procedimento de atualização do tamanho dos dados (Adaptado [11]) 19
Figura 17 Serviços de segurança no BLE (Adaptado [12]) 20
Figura 18 Cliente e Servidor GATT (Adaptado [11]) 21
Figura 19 Diagrama de Estado do GAP (Adaptado[11]) 24
Figura 20 Evento e intervalo de conexão[11] 25
Figura 21 Execução das Threads TI-RTOS (Adaptado [11]) 26
Figura 22 Arquitetura Icall [11] 28
Figura 23 Exemplo de uso da ICALL [11] 29
Figura 24 LAUNCHXL-CC2640R2F 30
Figura 25 Diagrama Funcional-CC2640R2F[13] 31
Figura 26 Caso de Uso 33
Figura 27 Diagrama de sequência da receção de sinal 34
Figura 28 Diagrama de sequência Ativar Entradas/Saídas digitais 35
Figura 29 Arquitetura geral do sistema 36
Figura 30 Sistema de aquisição e condicionamento do sinal 37

Figura 31 Características do serviço Simple Serial Socket Server 38
Figura 32 Fluxograma da aplicação 39
Figura 33 Fluxograma conversões ADC 39
Figura 34 Fluxograma processamento de mensagens 40
Figura 35 Sistema de aquisição e condicionamento do sinal 41
Figura 36 Ativação da notificação 42
Figura 37 Códigos da Receção dos dados 42
Figura 38 Resultado do ADCbufferContinuos pelo Code Composer Studio 43
Figura 39 main da aplicação 44
Figura 40 Implementação dos comandos 46
Figura 41 Formato de trama 47
Figura 42 Permissões Bluetooth 47
Figura 43 Ativação do Bluetooth no smartphone e acesso a localização 48
Figura 44 Exemplo do código da atividade Scaning 48
Figura 45 Exemplo do código da atividade conexão 49
Figura 46 Resultado de teste a uma onda quadrada 51
Figura 47 Comando Stop 51
Figura 48 Evolução da perda de dados nos canais de amostragem 52

Acrónimos

A2DP Advanced Audio Distribution Profile

ADC Analog to Digital Conversion

App Aplicação Móvel

AVRCP A/V Remote Control Profile

BLE Bluetooth Low Energy

CCCD Client Characteristic Configuration Descriptor

DMA Direct Memory Access

EOLC End Of Last Conversion

GAP Generic Access Profile

GATT Generic Attribute Profile

GFSK Gaussian frequency shift keying

GPIO General Purpose Input/Output

HFP Hands-Free Profile

HID Human Interface Device

HSP Headset Profile

I2C Inter-Integrated Circuit

IoT Internet of Things

ISR *Interrupt Service Routine*

L2CAP *Logical Link Control and Adaptation Layer Protocol*

PCB *Printed Circuit Board*

PC *Personal Computer*

PSIM *Physical Security Information Management*

RAM *Random Access Memory*

RTOS *Real Time Operating System*

SIG *Special Interest Group*

SPI *Serial Peripheral Interface*

SPP *Serial Port Profile*

TI *Texas Instrument*

Desenvolvimento de um sistema de aquisição de sinal sem fios para aplicação no desporto

Capítulo 1 – Introdução

1.1 Motivação e Contextualização

Atualmente a prática de exercício físico mais do que uma moda é um imperativo de saúde sendo por isso cada vez mais procurada por pessoas de todas as idades, indo desde as atividades mais ligeiras às mais exigentes e radicais. Em alguns casos esta atividade induz a prática desportiva de competição, os atletas amadores, que participam em provas competitivas e que têm uma prática desportiva mais regular e intensa. No extremo temos os atletas de competição que sendo profissionais ou não, competem a alto nível quer nacional ou internacionalmente. Em qualquer destes casos, a procura de tecnologias de auxílio à prática desportiva tem vindo a sofrer aumentos significativos. A sua utilização permite não só monitorizar dados relativos à prática desportiva propriamente dita, tais como distância, tempo e diferentes estatísticas, permitindo por outro lado a obtenção de parâmetros fisiológicos do atleta como ritmo cardíaco, pressão arterial, etc. Desta forma é possível aos atletas a monitorização das suas atividades e o controlo do seu desempenho, o que no caso dos atletas de competição passa também muitas vezes pela melhoria da técnica e outros parâmetros que visem um aumento do desempenho global.

Atualmente existe já uma variedade de dispositivos para monitorizar a atividade desportiva, desde os pedómetros e fitas de pulso que medem vários parâmetros de forma mais ou menos exatas, até às fitas de peito e dispositivos específicos que medem parâmetros de forma mais exata, permitindo aos atletas uma avaliação mais precisa do seu desempenho. Por outro lado, as várias modalidades de competições vão procurando medir novos parâmetros do desempenho dos atletas exigindo por isso novos e mais específicos dispositivos.

1.2 Objetivos

Esta dissertação visa o desenvolvimento de um sistema de aquisição de sinal sem fios baseado na tecnologia BLE para a aplicação no desporto. Este sistema auxilia os atletas a melhorar as suas performances, com as funcionalidades básicas de quantificar:

1. A força exercida: variação em volt do sinal proveniente do sensor;

2. O tempo de reação: tempo que leva o utilizador a responder ao acionamento do *led*, incluído no circuito de condicionamento do sinal, que dá início à recolha dos dados;

Para a realização desta dissertação foi necessário estudar um conjunto de técnicas e métodos, quer na realização de simulações recorrendo ao PSIM e programação do micro- controlador. A nível técnico, a implementação do projeto contempla:

1. Um sistema de aquisição de sinal baseado em microcontrolador;

2. *Software* para aquisição e transmissão dos dados por Bluetooth para um dispositivo móvel;

3. *App* para processamento e visualização num dispositivo móvel.

A criação de um sistema de aquisição em colaboração com a aplicação Android permite quantificar a força exercida ao longo de um treino (aquisição continuo de sinal) e, ainda, calcular o tempo de reação, principais funcionalidades a concretizar neste projeto.

1.3 Estrutura da Dissertação

No presente capítulo é feita a contextualização do trabalho realizado. São, também, descritas as motivações e os objetivos que conduziram ao desenvolvimento deste trabalho. No segundo capítulo é descrito o estado da arte dos princípios que se encontram inerentes a esta dissertação, como: trabalhos relacionados, Bluetooth, sensor piezoresistivo, RTOS e o hardware. No terceiro capítulo são apresentados os requisitos, casos de uso e diagramas de sequência fazendo uma análise geral do problema. No capítulo quatro é apresentado o desenvolvimento do projeto, onde se encontram as informações referentes a especificação e implementação do sistema, a descrição do *software* e os passos que foram tomados ao longo desta investigação, de forma a que fosse possível atingir os objetivos estipulados. No quinto capítulo é feita uma reflexão acerca das experiências realizadas e do trabalho desenvolvido. No último capítulo, capítulo seis, são descritas as conclusões do trabalho desenvolvido no âmbito da dissertação e as perspetivas futuras para o mesmo.

Capítulo 2 – Estado da Arte

A tecnologia é um recurso em constante evolução, que proporciona cada vez mais um maior bem-estar e uma maior utilidade em quase todas as atividades desenvolvidas pelo Homem.

Encontra-se também no desporto, permitindo a monitorização dos atletas, quando desde dispositivos básicos e comuns, até complexos sistemas que avaliam o de- sempenho dos atletas de competição. Permite assim obter melhores resultados a nível das classificações dos atletas, nos treinos desportivos e no estudo das aptidões e das capaci- dades de cada atleta, de forma a analisar os pontos que devem ser melhorados e quais os planos de trabalho mais indicados para cada modalidade ou atleta.

Existem no mercado exemplos de produtos tecnológicos que permitem a monitorização de treino para ajudar neste objetivo. Relativamente aos produtos que se encontram rela- cionados com o caso de uso aqui proposto, destacam-se produtos como Corner[1], Hykso Trackers[2] e Impactwrap[3].

2.1 Trabalhos Relacionados

No contexto de estudo desta dissertação é importante também encontrar projetos que se possam assemelhar a este de modo a perceber os processos usados na implementação, ou até os materiais escolhidos para as diferentes aplicações.

2.1.1 *Low Cost Wireless Data Acquisition System for Multi- sensor Applications*

Um dos projetos encontrados numa fase preliminar do desenvolvimento desta dissertação relacionava-se com sistemas de irrigação automatizados para fins agrícolas. A escassez de recursos hídricos e a crescente procura da produção de alimentos foram fatores que impulsionaram o seu desenvolvimento.

Neste projeto foram diversas as ferramentas e técnicas utilizadas. Os nós do sensor e o formato do *frame* dos dados do sensor foram construídos usando um microcontrolador Atmega328. O X-Bee concede a conectividade sem fio entre os nós dos sensores e para construção do nó coordenador foi usado um ATmega2560, que por sua vez comunica com o LabVIEW[4].

A Figura 1 representa o diagrama de bloco do sistema: para o nó sensor e nó coordenador.

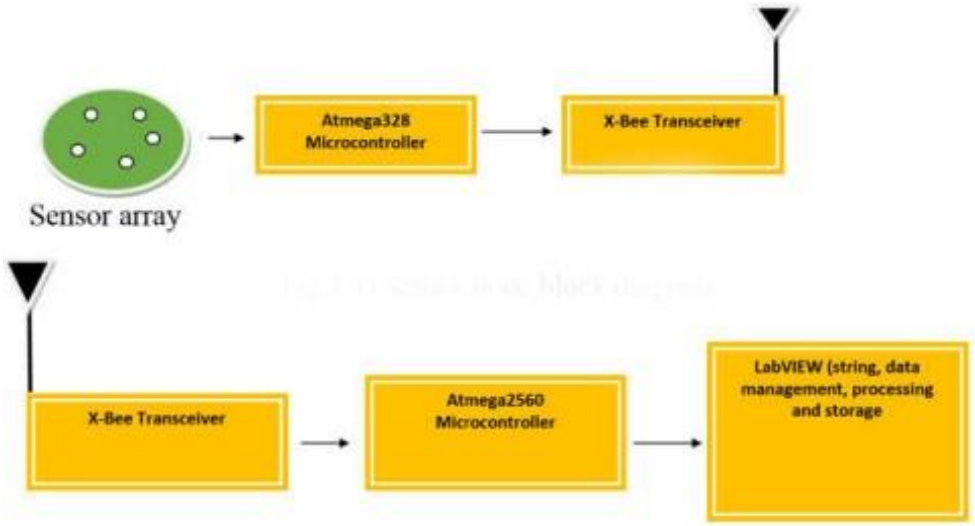

Figura 1 - Diagrama de bloco do sistema[4]

2.1.2 *Wireless Data Acquisition System for IoT Applications*

O outro exemplo de projeto de implementação de um sistema aquisição de sinal relaciona-se com o setor da saúde. Trata-se de um sistema capaz de comunicar com um cinto torácico, a fim de obter o valor da frequência cardíaca e os dados do acelerómetro de um relógio EZ Chronos. O principal objetivo deste projeto seria implementar um sistema incorporado, que pudesse ser usado em diversas aplicações médicas de assistência/suporte de vida[5].

A Figura 2 representa o esquema experimental da plataforma em que o projeto se baseia.

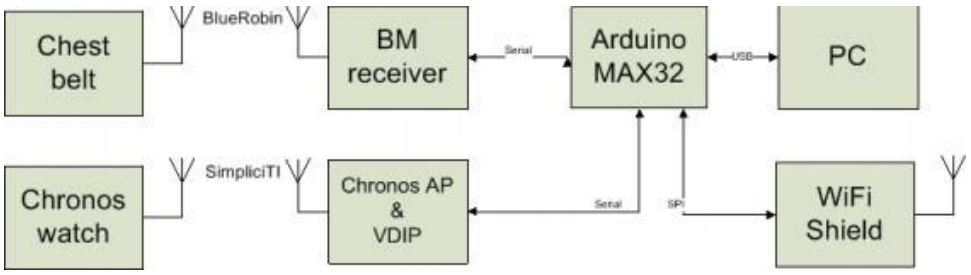

Figura 2 Diagrama de bloco do sistema[5]

2.1.3 Impact Wrap

O Impactwrap[3], Figura 3, lançado no mercado em 2018, já possui a sua plataforma para ginásios disponível. É um rastreador *fitness* portátil que capta a potência e a frequência com que os atletas aplicam os golpes nos treinos. O sistema de aquisição pro- posto nesta dissertação difere do Impact Wrap já que o impacto é aplicado, diretamente, no sensor, enquanto o Impact Wrap usa um acelerómetro que apenas mede a aceleração do movimento.

Figura 3 Impact Wrap

2.1.4 Corner

O Corner[1] é uma tecnologia já disponível no mercado, fabricado por Athletec, anunciada em janeiro de 2016, que consiste em encaixar em cada uma das luvas do atleta, Figura 4, um pequeno dispositivo com um acelerómetro de três eixos. O dispositivo rastreia tipos de soco (soco, cruz, gancho, uppercut), velocidade, potência, taxa de acer- tos/falta, blocos, combinações, taxa de soco e ritmo de um atleta durante um treino de boxe.

Figura 4 Corner

2.1.5 Hykso Trackers

O Hykso Trackers[2], Figura 5, é uma tecnologia que já se encontra disponível no mercado, anunciada em junho de 2016, com as funcionalidades de cálculo da quantidade de golpes aplicados no saco, bem como a velocidade e o tipo de golpes durante um treino. É usado dentro das luvas do atleta e é constituído por dois acelerómetros independentes e um giroscópio para a captura dos movimentos 3D completos a uma taxa de 1000 vezes por segundo.

Figura 5 Hykso Trackers

2.2 Bluetooth

O Bluetooth é um sistema de comunicação sem fios que permite a troca de dados entre dispositivos móveis ou fixos, desenvolvido pelo SIG (Special Interest Group) em 1998 e lançada oficialmente em 1999[14].

As redes Bluetooth (designadas de *piconets*) usam um modelo mestre/escravo para controlar quando e onde os dispositivos podem enviar os dados. Neste modelo, um único dispositivo mestre pode ser conectado a até sete dispositivos escravos diferentes. Qualquer dispositivo escravo na *piconet* só pode ser conectado a um único mestre, como se encontra representado na Figura 6.

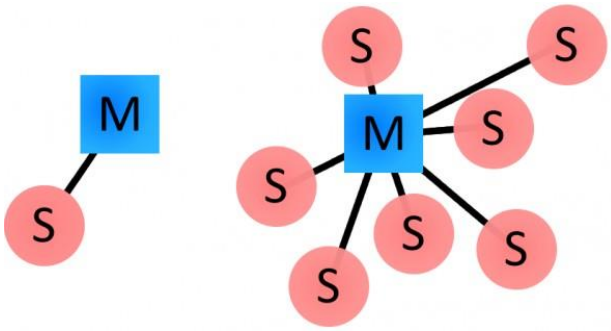

Figura 6 Exemplos de topologias de piconet mestre/escravo[6]

No sistema de tecnologia Bluetooth o dispositivo pode ser emissor(mestre) ou recetor(escravo) de dados dependendo da aplicação que se pretende. Atualmente existem disponíveis no mercado vários tipos de

Bluetooth, que possibilitam dispositivos tais como PC, rádios, auriculares, telemóveis a comunicarem sem fios. Cada dispositivo é equipado por um micro-circuito (trans-receptor) que transmite e recebe informações numa frequência de 2,4 GHz, sendo que os dados podem ser transmitidas a uma velocidade de até 1 megabit por segundo ou 2 megabits por segundo na mais recente versão dessa tecnologia (Bluetooth 5).

As informações transmitidas por Bluetooth são divididas em pacotes e, cada pacote é transmitido num dos 79 canais Bluetooth. Cada canal tem uma largura de banda de 1MHz. O primeiro canal começa em 2402 MHz e continua até 2480 MHz em passos de 1 MHz.

De acordo com a potência de transmissão, os dispositivos Bluetooth são classificados em categorias que variam de 1mW a 10mW como se encontra apresentado na Figura 7.

Device power class	Power Emitted	Range
Class 1	100mW	~ 100 m
Class 2	2.5mW	~10 m
Class 3	1mW	~1 m

Figura 7 Classes do Bluetooth[7]

Para efetuar uma conexão Bluetooth entre dois dispositivos é necessário um processo de várias etapas que envolve três principais fases:

1. Consulta - Se *a priori* dois dispositivos Bluetooth não se conhecem, ou seja, se nunca houve uma conexão entre eles, é preciso executar uma ligação de modo a conhecer o endereço um do outro. O dispositivo envia uma solicitação, a todos os outros dispositivos que estão no seu alcance, que recebe esse pedido e envia o seu endereço e outras informações.

2. *Paging (Connecting)* - Para que se estabeleça uma conexão entre dois dispositivos, cada dispositivo precisa conhecer o endereço um do outro, a este processo de formar conexão designa-se *paging*.

3. Conexão - Após um dispositivo ter concluído o processo de *paging*, ele entra no estado da conexão. Quando conectado, um dispositivo pode permanecer em um dos seguintes modos de consumo de energia:

   (a) Modo Ativo - Neste modo o dispositivo pode transmitir ou receber dados;

   (b) Modo Sniff - Neste modo o dispositivo é menos ativo, modo de economia de energia. O dispositivo escuta transmissões em um intervalo definido (por exemplo, a cada 100ms);

(c) Modo hold - Neste modo o dispositivo fica inativo durante um período de tempo definido, e de seguida retorna ao estado ativo quando esse intervalo é ultrapassado;

(d) Modo Park - Neste modo o dispositivo fica inativo até receber ordem do mestre para ativar.

Quando dois dispositivos Bluetooth se conectarem pela primeira vez, eles compartilham os seus endereços, nomes e perfis, e geralmente essas informações são guardadas para futuras conexões. Dispositivos ligados estabelecem automaticamente, uma conexão sempre que estão próximos o suficiente[6].

2.2.1 Camada Lógica

Esta é a camada responsável por selecionar um formato de transporte mais adequado para cada canal, consoante a(s) taxa(s) de fonte dos canais lógicos. O formato de transporte é selecionado relacionando o conjunto de combinação de formatos de transporte, definido a partir do controlo de admissão para cada conexão.

Segundo Labiod et al.[8], a camada lógica é constituídas por três entidades lógicas:

- A entidade lógica MAC-b gere o canal de transmissão;
- A entidade lógica MAC-c / sh gere os canais comuns e compartilhados: canal de *paging*, canal de acesso ao link direto, canal de acesso aleatório, canal de pacotes comum de *uplink* e o canal compartilhado de *downlink*;
- A MAC-d é responsável por gerir os canais dedicados no modo conectado.

Os pacotes transferidos durante a comunicação Bluetooth são transportados através da camada MAC. Os pacotes envolvidos na comunicação possuem uma estrutura específica, tal como representa a Figura 8, composta pelo *"Access Code"*, *"Header"* e *"Payload"*.

- *Access Code:* o código de acesso é usado para sincronização de tempo, compensação de deslocamento, *paging* e consulta. Existem três tipos diferentes de código de acesso: Código de acesso ao canal (CAC), Código de acesso ao dispositivo (DAC) e Código de acesso à consulta (IAC). O código de acesso (CAC) ao canal identifica uma *piconet* exclusiva enquanto o DAC é usado para *paging* e suas respostas. O IAC é usado para fins de consulta.
- *Header:* O cabeçalho contém informações para reconhecimento de pacotes, nu- meração de pacotes para reordenação de pacotes fora de ordem, controle de fluxo, endereço escravo e verificação de erro do cabeçalho.

- *Payload:* a carga útil do pacote pode conter campo de voz, campo de dados ou ambos. Possui um campo de dados, a carga útil também conterá um cabeçalho de carga útil.

Figura 8 Estrutura dos pacotes[8]

2.3 Bluetooth Low Energy

O Bluetooth Low Energy, também muitas das vezes referido como Bluetooth Smart, é um subconjunto do Bluetooth clássico, e foi introduzido como parte da especificação principal do Bluetooth 4.0. Embora haja alguma sobreposição com o Bluetooth clássico, o BLE tem uma origem diferente e foi iniciado pela Nokia como um projeto interno chamado Wibree antes de ser adotado pelo Bluetooth SIG[9]. O BLE é uma tecnologia wireless que se encontra a emergir no âmbito das comunicações de short-range (pequena distância).

A expansão do Bluetooth através da sua implementação em dispositivos móveis, tais como os telemóveis, os computadores, entre outros, e a emergência do conceito da IoT (Internet of Things) permitiram um aumento do interesse no desenvolvimento e expansão do BLE já que seria possível uma implementação Bluetooth que tem a capacidade de economizar recursos energéticos, o que lhe confere uma vantagem sobre o Bluetooth clássico.

Tal como as primeiras versões do Bluetooth, também o BLE é composto por duas camadas principais: a camada do controlador e a camada do *host* (anfitrião). Enquanto que a camada do controlador contempla a camada de ligação e a camada física, a camada do *host* funciona num processador de aplicações e inclui ainda uma camada superior funcional.

O suporte para Bluetooth Low Energy (subconjunto do BT 4.0) está disponível na maioria das principais plataformas e sistemas operativos que se encontram no mercado (iOS, Andorid, Mac OS X, Windows, GNU/Linux).

O Generic Access Profile (GAP), dita como os dispositivos Bluetooth podem (ou não) interagir uns com os outros. O GAP controla as conexões, descoberta de dispositivos e estabelecimento de segurança, para garantir a interoperabilidade e permitir a troca de dados entre dispositivos. O GAP define várias funções para dispositivos, sendo os dispositivos centrais, tais como, telemóvel ou tablet, e os dispositivos periféricos tal como *smartband* que são emparelhados com um dispositivo central, como os principais conceitos.

Na Figura 9 encontra-se representado um exemplo de como são realizadas os pedidos e respostas por parte dos dispositivos centrais e periféricos.

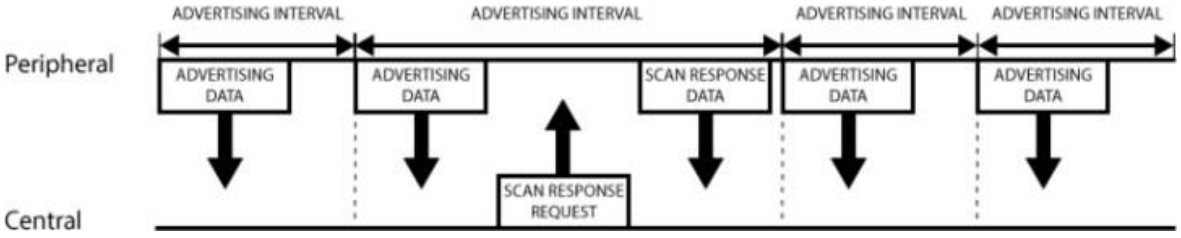

Figura 9 Funcionamento entre as cargas do protocolo GAP[9]

O dispositivo periférico define um período específico de envio de carga útil de dados, e se esse intervalo for ultrapassado esse mesmo periférico volta a transmitir o pacote.

Como já referido anteriormente, se um dispositivo central estiver interessado num *load*, este pode solicitá-lo e o periférico responde com esses dados, tal como representa a Figura 9.

Depois de uma conexão entre dois dispositivos Bluetooth ser estabelecida, o GATT define como são transferidos os dados entre o dispositivo periférico e o central. O GATT faz uso de um protocolo de dados genérico chamado ATT (Attribute Protocol), que é usado para armazenar serviços, características e dados relacionados numa tabela de consulta simples, usando IDs de 16 bits para cada entrada de dados nessa mesma tabela[9].

O relacionamento servidor/cliente permite uma melhor compreensão do conceito GATT.

O dispositivo periférico é conhecido como Servidor GATT, que contém dados de consulta ATT. As definições de serviço e características são designados de Cliente GATT (telemóvel / tablet), que enviam solicitações ao servidor[9].

Todas as transações são iniciadas pelo dispositivo mestre, o Cliente GATT, que recebe resposta do dispositivo escravo, o Servidor GATT, como se encontra representado na Figura 10.

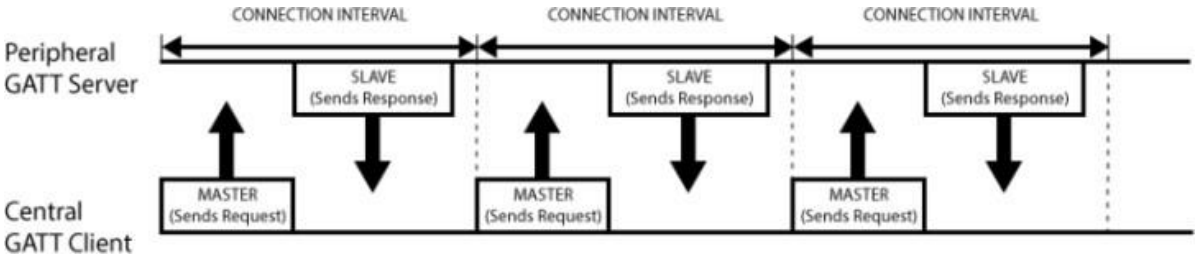

Figura 10 Troca de dados entre um dispositivo periférico e um central[9]

As trocas de dados GATT no BLE são baseadas em objetos de alto nível. Estes objetos podem ser *Profiles*, *Services* ou *Characteristics*, como está ilustrado na Figura 11.

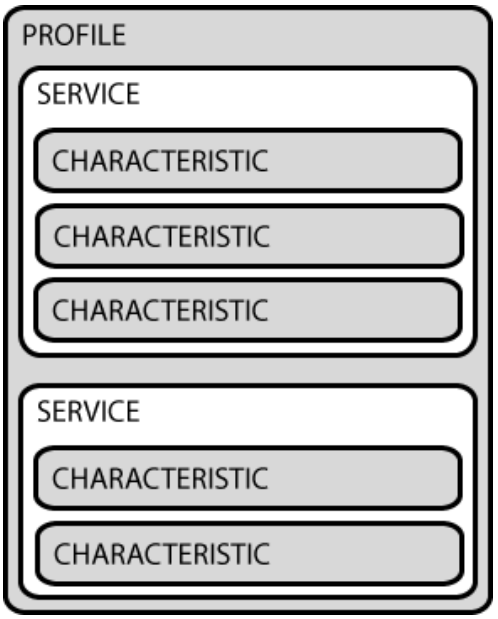

Figura 11 Organização da base de dados do Bluetooth BLE[9]

Um perfil é uma coleção pré-definida de serviços compilada pelo Bluetooth SIG ou pelos designers de periféricos[9].

Os serviços são usados para dividir em entidades lógicas, e contêm partes especificas de dados chamadas características. Um serviço pode ter uma ou mais características, e cada serviço distingue-se um do outro por meio de um ID de 16 bits para Serviços BLE adotados oficialmente ou 128 bits para serviços personalizados[9].

As caraterísticas são o principal ponto onde um utilizador irá interagir com um periférico BLE, para além disso estas são usadas para enviar dados de volta para o Servidor GATT[9].

Este projeto contempla a comunicação entre duas plataformas: um sistema de aquisição de sinal e um dispositivo externo de receção. Pela complexidade do processo que se estabelece entre a fonte da informação e o recetor da mesma é importante a utilização de uma tecnologia como o Bluetooth BLE pelo facto de a sua principal característica ser a economia de energia e também por se tratar de um o sistema que processará e efetuará transmissão contínua de dados com uma data rate elevada, conseguindo assim uma comunicação fiável e estável. A implementação do BLE permitirá que a informação gerada no saco de boxe (emissor) seja reencaminhada para um dispositivo, que pode ser, por exemplo, um computador ou um smartphone, que terá o papel de recetor da informação. A informação recebida será depois monitorada e aplicada na melhoria das técnicas de treino para que, mais tarde, se possam verificar progressos nos desempenhos dos atletas que utilizem o sistema proposto.

Neste projeto será utilizada a versão 4.2 do Bluetooth BLE, a mais recente lançada até este momento.

2.3.1 Bluetooth 4.2

Apresentado no final de 2014, o Bluetooth 4.2 trouxe diferenças importantes. Entre outros protocolos, a versão tem pleno suporte ao IPv6 para tornar a tecnologia ainda mais relevante para a Internet das coisas (IoT): câmaras de segurança, lâmpadas inteligentes, termostatos e outros dispositivos domésticos podem usar a tecnologia de modo otimizado para comunicação no mesmo ambiente ou para acesso à Internet.

O Bluetooth 4.2 também usa criptografia do tipo FIPS (mais avançado) nas conexões e tem um controlo mais rigoroso da segurança, assegurando que apenas dispositivos devidamente autorizados se conectam a outros.

A velocidade de transferência de dados permanece padronizada em 24 Mb/s, mas o Bluetooth 4.2 suporta tráfego de dados maior, ou seja, os dispositivos podem enviar e receber mais dados ao mesmo tempo.

2.3.2 Camada Física

A camada de ligação é composta por um circuito de comunicações análogas responsável por traduzir "símbolos" digitais que são transmitidos. Trata-se da camada mais baixa da "pilha" de protocolos e fornece, ainda, serviços à camada de ligação.

Esta camada usa ondas rádio da banda ISM (Industrial, Científica e Médica) de 2.4 GHz por forma a comunicar e dividir essa mesma banda de frequências em 40 canais, cada canal com 2 MHz, espaçados dos 2.4000 GHz até aos 2.4835 GHz, a começar nos 2402 MHz, tal como mostra a Figura 12.

Os 40 canais criados encontram-se distribuídos como sendo "Advertising Channel Usage" e "Data Channel Usage". O primeiro tipo de canais contempla as funcionalidades de: descoberta de dispositivos, estabelecimento de conexões e transmissões *broadcast*. Enquanto que o segundo tipo de canais diz respeito a capacidades de comunicações bidirecional entre dispositivos conectados e adaptação da frequência de *hopping* usada para eventos de conexão subsequentes.

O BLE transmite dados a uma taxa de 1 Mbps. No entanto, a sobrecarga do protocolo limita a taxa de transferência para um máximo abaixo dos 1Mbps. Por outro lado, o BLE usa o Gaussian frequency-shift keying (GFSK), pelo qual os dados são filtrados recorrendo a um filtro de Gauss antes de serem usados para alterar a frequência portadora, a fim de tornar as transições de frequência mais ligeiras[10].

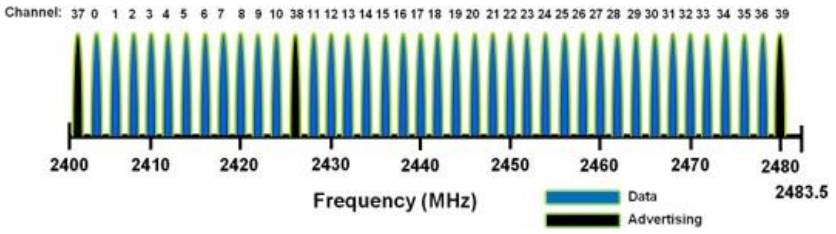

Figura 12 Distribuição dos canais na banda ISM[10]

2.3.3 Camada Ligação

A camada de ligação é a que contacta diretamente com a camada física. Esta camada é responsável por publicitar, fazer *scan*, e criar/manter as conexões já estabelecidas.

Para que estas responsabilidades sejam cumpridas, a camada de ligação deve preocupar-se com os canais, referidos anteriormente, os pacotes e a conexão, e descoberta de procedimentos.

Existem alguns conceitos-chave envolvidos na comunicação/ligação que se estabelece com o BLE. *Advertising channels, advertising packet, advertiser, advertising event, initiator, connection request,* são alguns dos conceitos importantes para que a comunicação BLE seja estabelecida e, melhor compreendida.

- ***Advertising Channels:*** são canais por onde os *advertisng packets* são enviados durante a comunicação;
- ***Advertising Packets:*** são os pacotes de informação que são envidas ao longo da comunicação, identificados com um código de acesso de 32 bits gerado aleatoriamente;
- ***Advertiser:*** dispositivos que enviam os *advertising packets*;
- ***Advertising Event:*** intervalos entre os quais os *advertising packets* são transmitidos ao longo dos *Advertising Channels*;
- ***Initiator:*** o segundo dispositivo implicado na comunicação BLE bidirecional;
- ***Connection Request:*** mensagem que cria uma conexão ponto-a-ponto entre os dois dispositivos presentes na comunicação BLE bidirecional.

No sistema de comunicação BLE, ao nível da camada de ligação, Link Layer, são definidos dois papéis: o master, mestre, e o *slave*, escravo. O mestre e o hospedeiro são o mesmo que o iniciador e o *advertiser* durante a criação de uma conexão, respetivamente. No caso do mestre, este pode estabelecer e gerir várias conexões simultaneamente com diferentes escravos. Por outro lado, um escravo pode conectar-se apenas a um mestre. A conexão que se estabelece entre um mestre e diversos escravos é chamada de *piconet*.

Com o objetivo de economizar energia, os escravos estão no modo de suspensão por padrão e periodicamente "acordam" para escutar possíveis receções de pacotes do mestre. É o mestre que determina quando os escravos devem acordar para escutar as possíveis receções de pacotes do mestre, coordena o acesso médio (camada MAC) recorrendo ao esquema de Time Division Multiple Access (TDMA) e, fornece ao escravo as informações necessárias para o algoritmo de salto de frequência (frequency hopping) e para a supervisão da conexão [12].

2.3.4 Host Controller Interface (HCI)

Acamada Host Controller Interface (HCI) é uma pequena camada que realiza o transporte de comandos e eventos que fazem parte do mestre, e do controlador da *stack* de protocolos Bluetooth. Numa aplicação de processador de rede pura, a camada HCI é implementada por meio de um protocolo de transporte como SPI (Serial Peripheral Interface) ou UART (Universal asynchronous receiver-transmitter). Em projetos que incorporam microcontroladores wireless (sem fios), como simple_peripheral ou simple_np, a camada HCI é implementada por meio de chamadas da função e retornos de chamada no microcontrolador sem fio. Camadas e eventos, como ATT e o GAP, atravessam as camadas superiores da *stack* de protocolos através da camada HCI até que chegam ao controlador. Da mesma forma, o controlador envia dados e eventos recebidos para o mestre e para as camadas superiores através do HCI.

O TI BLE-Stack suporta uma configuração de processador de rede (host_test) que permite que uma aplicação que se encontra em execução, num determinado microcontrolador externo, interaja com a TI BLE-Stack. O processador de rede pode aceitar todos os comandos HCI Low Energy (LE); Apesar de tudo, como o mestre e o controlador BLE coabitam no microcontrolador sem fios, alguns dos eventos dos comandos HCI serão consumidos pelo mestre TI BLE.

2.3.5 Logical Link Control and Adaptation Layer Protocol (L2CAP)

O L2CAP usado no BLE é um protocolo que resulta da otimização e simplificação do clássico Bluetooth L2CAP. Esta camada localiza-se na parte superior da camada HCI. No BLE, o principal objetivo do L2CAP é proliferar os dados provenientes do protocolo ATT (Attribute Protocol), SMP (Simple Management Protocol), os quais se encontram numa camada de protocolos superior à do L2CAP, e procede à sinalização de controlo da Camada de Ligação. Os dados provenientes destes protocolos são tratados pelo L2CAP, o qual faz uso de uma abordagem *best-effort*, onde o protocolo procura encontrar os dados a taxa e latência variáveis o que pode não ir de encontro à melhor qualidade dos dados. Para além da abordagem *best-effort* o L2CAP não faz uso de mecanismos de retransmissão e controlo de fluxo, que estão disponíveis noutras versões do Bluetooth, nem usa os recursos de segmentação e remontagem, já que os protocolos tratados pelo L2CAP fornecem unidades de dados que se ajustam ao tamanho máximo da sua carga útil, o que equivale a 23 bytes, especificamente no caso do BLE[12].

Termo	Descrição
Canal L2CAP	A conexão lógica entre dois pontos extremos em dispositivos compatíveis, caracterizados pelos seus identificadores de canal (CIDs)
SDU ou L2CAP SDU	Service Data Unit: é um pacote de dados que o protocolo L2CAP permuta com uma camada superior e transporta, de forma transparente, sobre um canal L2CAP a partir do uso de procedimentos específicos
PDU ou L2CAP PDU	Protocol Data Unit: um pacote de dados que contém campos de informação, informação de controlo e informação acerca da camada superior do protocolo L2CAP
Maximum Transmission Unit (MTU)	O tamanho máximo de *payload* de dados, em octetos, que a entidade da camada superior pode aceitar (o mesmo que se verifica com o MTU que corresponde ao tamanho máximo SDU)
Maximum PDU Payload Size (MPS)	O tamanho máximo de *payload* de dados, em octetos, que a entidade da camada do L2CAP pode aceitar (o mesmo que se verifica com o MPS que corresponde ao tamanho máximo PDU)

Figura 13 Terminologia Geral do L2CAP (Adaptado [11])

O L2CAP também é baseado em canais como mostra a Figura 13. Cada ponto final de um canal L2CAP é referido por um identificador de canal (CID). No caso do CID do L2CAP é alocado dinamicamente para identificar o link lógico e o terminal local. O terminal local deve estar no intervalo de 0x0040 a 0xFFFF [12].

A Figura 14 mostra arquitetura de blocos do protocolo L2CAP a qual apresenta os fluxos de dados/pacotes e os controlos que se estabelecem na cadeia do protocolo.

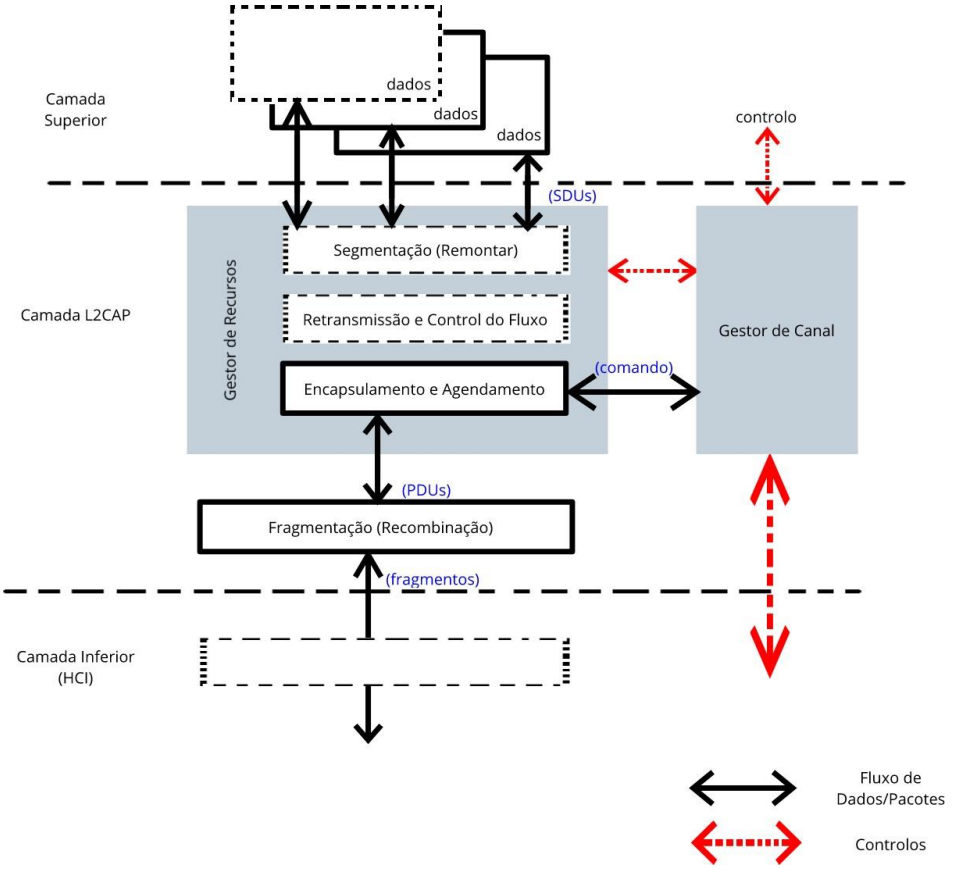

Figura 14 Arquitetura L2CAP (Adaptado [11])

2.3.6 Maximum Transmission Unit (MTU)

A *stack* BLE suporta fragmentação e a recombinação de PDUs L2CAP na camada de ligação. O suporte à fragmentação permite que o L2CAP e os protocolos de nível superior criados sobre o L2CAP, como é o exemplo do ATT, façam uso de tamanhos de carga úteis superiores e diminuam a sobrecarga relacionada com as transações de dados de maiores dimensões. Quando a fragmentação é usada, os pacotes maiores são divididos em vários pacotes da camada de ligação e reagrupados pela camada de ligação do dispositivo de nível equivalente[11], tal como se apresenta na Figura 15.

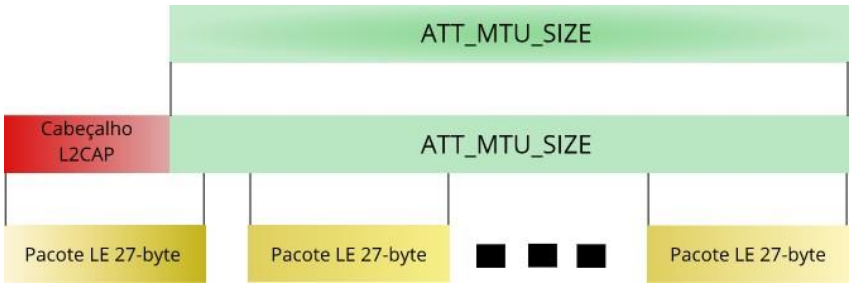

Figura 15 Pacote L2CAP mais Protocolo de Atributo (Adaptado[11])

2.3.7 Data Length Extension (DLE)

O recurso de Data Length Extension (DLE) fornece a capacidade para que o controlador LE envie *packet data units* (PDUs) do canal de dados com cargas úteis que podem atingir os 251 bytes de dados da aplicação, enquanto estiver em estado de conexão. Para além disso, pode ser negociado um novo tamanho de PDU pelos dois lados, a qualquer momento durante uma conexão. Em versões anteriores, a maior carga útil do canal de dados do controlador era de apenas 27 bytes. Este aumento de carga permite que a taxa de dados cresça em cerca de 2,5x, em comparação com os dispositivos Bluetooth Core Specification das versões 4.0 e 4.1.

Depois de uma conexão ser formada, os controladores LE do dispositivo podem usar as PDUs de controlo LL_LENGTH_REQ e LL_LENGTH_RSP para negociar um tamanho de carga útil superior para as PDUs do canal de dados. A atualização do comprimento de dados pode ser iniciada pelo mestre ou executada de forma autónoma pelo controlador. O mestre, ou o escravo podem iniciar este procedimento.

Após a conclusão do procedimento de atualização do comprimento dos dados, os dois controladores selecionam um novo comprimento de dados tendo em conta dois parâmetros: tamanho da PDU e tempo. Esta seleção é feita de acordo com a Figura 16. O maior tamanho suportado pelos controladores, local e remoto, é escolhido, enquanto que o tempo é levado em consideração para suportar as diferentes taxas de dados [11].

Recurso de extensão de comprimento de pacote de dados LE suportado	Parâmetros com nomes terminados em Octetos		Parâmetros com nomes terminados em tempo (µs)	
	Mínimo	Máximo	Mínimo	Máximo
Não	27	27	328	328
Sim	27	251	328	2120

Figura 16 Tamanho e tempo do procedimento de atualização do tamanho dos dados (Adaptado [11])

2.3.8 Camada de Gestão de Segurança

Com o papel de proteger as informações que são transferidos entre dispositivos a camada de segurança fornece diversos serviços de segurança. A maior parte destes serviços de segurança pode ser classificada em dois modos: LE Modo de Segurança 1, ou LE Modo de Segurança 2. Estes modos de segurança estão associados à camada de ligação e camada ATT, respetivamente.

Como forma de garantir a segurança das comunicações, o BLE suporta um mecanismo chamado recurso de privacidade. Este mecanismo permite que um dispositivo use endereços privados e os altere com frequência. Para além de que reduz a ameaça pela qual um "inimigo" pode encontrar um dispositivo BLE. Os endereços privados são criptografados a partir do endereço público do dispositivo, que pode ser desenvolvido por um dispositivo confiável que possua a chave de criptografia correspondente.

Cada modo de segurança encontra-se dividido por diferentes níveis, que expressam requisitos quanto ao tipo de conexão que deve ser usada. O processo de conexão permite aos dispositivos gerar e distribuir

os principais materiais para efetivar a comunicação com outros. A Figura 17 resume os serviços de segurança e o tipo de emparelhamento (se houver) exigido por cada modo e nível de segurança [12].

		Emparelhar	Encriptação	Integridade dos dados	Camada
LE Modo de Segurança 1	Nível 1	Não	Não	Não	Camada de Ligação
	Nível 2	Sem autenticação	Sim	Sim	
	Nível 3	Autenticado	Sim	Sim	

Figura 17 Serviços de segurança no BLE (Adaptado [12])

A conexão compreende três fases. A primeira fase, ambos os dispositivos conectados anunciam as suas capacidades de input/output e baseando-se nessas informações selecionam o método mais adequado para utilizar na segunda fase.

A segunda fase tem como objetivo gerar a *Short-Term Key* (STK), ou seja, Chave de curto prazo, a partir do método escolhido na primeira fase. Esta chave será usada na terceira fase para garantir a distribuição do material da chave dos dispositivos. Na segunda fase, os dispositivos de emparelhamento concordam com uma *Temporary Key* (TK), ou Chave Temporária. A TK e os valores aleatoriamente gerados por cada um dos dispositivos de conexão são usados para que os dois dispositivos possam obter a STK.

Na terceira fase, cada *endpoint* da conexão pode distribuir para o outro *endpoint* até três chaves de 128 bits chamadas *Long-Term Key* (LTK), a conexão *Signature Resolving Key* (CSRK) e a *Identity Resolving Key* (IRK). A LTK é usada para gerar a chave de 128 bits usada para criptografia e autenticação da camada de ligação. A CSRK é usada para a assinatura de dados realizada na camada ATT. Por último, a IRK, é usada para gerar um endereço privado com base no endereço de um dispositivo público.

Por fim, o *Security Manager Protocol* (SMP) realiza a troca de mensagens das três fases de conexão, o qual opera sobre um canal L2CAP fixo.

Em termos de segurança, uma das vulnerabilidades que existe atualmente no BLE é o facto de que nenhum dos métodos de conexão se encontrar contra a escuta passiva. Portanto, um "inimigo" que obtenha as mensagens de conexão pode determinar tanto a LTK, a CSRK ou a IRK. No entanto, uma versão futura do BLE poderá recorrer criptografia de curva elíptica e trocas de chaves públicas Diffie-Hellman de forma a resolver o problema de vulnerabilidade [12].

2.3.9 Attribute Protocol (ATT)

O ATT (Attribute Protocol) define a comunicação realizada entre dois dispositivos, os quais ocupam as funções de servidor e cliente, respetivamente, acima de um canal L2CAP dedicado. A camada ATT fornece a capacidade de um dispositivo partilhar determinados dados e atributos. Um desses atributos é a estrutura

de dados que armazena as informações acerca do protocolo que se encontram sob a gestão do GATT (Generic Attribute Profile).

A função de cliente ou servidor é determinada pelo protocolo GATT e é independente da função escrava ou mestre. O cliente tem a capacidade de aceder aos atributos do servi- dor a partir do envio de solicitações, o que desencadeia o envio de mensagens de resposta do servidor. Para que este processo seja mais eficiente, o servidor também pode enviar para o cliente dois tipos de mensagens não solicitadas que contêm atributos: notificações não confirmadas; e indicações, as quais exigem que o cliente envie uma confirmação. Para além deste método, o cliente também pode enviar comandos para o servidor para gravar valores de atributo. As transações de solicitação/resposta e indicação/confirmação seguem o método de *stop and wait*, isto é a solicitação só é concretizada após a confirmação ser recebida [12].

2.3.10 Generic Attribute Profile (GATT)

O GATT (Generic Attribute Profile) possui um papel semelhante ao protocolo ATT relacionando-se com a comunicação entre dois dispositivos (master e slave), como o apre- sentado na Figura 18. É esta camada que define uma *framework* a partir do ATT, protocolo acima referido, para a descoberta de serviços e a troca de características de um dispositivo para outro. Ao nível da pilha do protocolo BLE, os dados são transmitidos e guardados entre os dispositivos em forma de características, as quais são conjuntos de dados que incluem um valor e propriedades[11]. Por exemplo, um servidor que executa um serviço de 'sensor de temperatura' pode considerar uma característica de 'temperatura' que usa um atributo para descrever o sensor, outro atributo para armazenar valores de medição de temperatura e outro atributo para especificar as unidades de medição[12].

Segundo a Texas Instruments[11], do ponto de vista do GATT, quando dois dispositivos se encontram conectados cada um deles tem um papel entre:

- O servidor GATT é o dispositivo que contém a base de dados de características que está a ser lida ou gravada por um cliente GATT, ou;
- O cliente GATT é o dispositivo que está a ler ou gravar dados do, ou para o servidor GATT.

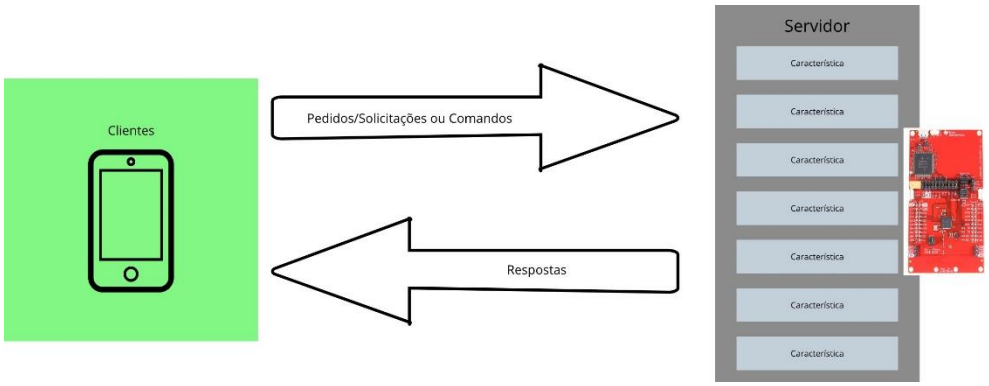

Figura 18 Cliente e Servidor GATT (Adaptado [11])

As funções GATT de cliente e servidor são independentes das funções GAP de periférico e central. Um escravo pode ser um servidor GATT e uma central pode ser um mestre GATT.

As características são grupos de informações e dados. Estas são uma peça importante no protocolo GATT, já que são estas que organizam e usam os atributos como sendo valores de dados, propriedades e informação de configuração. Normalmente, na pilha BLE as características são constituídas por atributos como:

- Valor da característica: Valor dos dados da característica;
- Declaração de característica: Trata-se descritor que armazena as propriedades, local e tipo do valor da característica;
- Configuração da característica do cliente: Trata-se de uma configuração que permite ao servidor GATT configurar a característica que deve ser notificada ou indicada;
- Descrição do Utilizador da Característica: *String* ASCII que descreve a característica.

Os atributos, por sua vez, são armazenados no servidor GATT numa tabela de atributos. Para além do valor, os atributos possuem algumas propriedades específicas:

- Identificador: Índice do atributo na tabela de identificadores;
- Tipo: Esclarece o que representam os dados do atributo, ou também conhecido como UUID, do português identificador exclusivo universal;
- Permissões: Define a forma como um dispositivo cliente GATT pode aceder ao valor de um atributo [11].

2.3.11 Generic Access Profile (GAP)

No nível mais alto da *stack* BLE, o GAP (Generic Access Profile) especifica as funções, modos e procedimentos para a descoberta de dispositivos e serviços. Esta camada possui um papel mais relacionado com a conexão quando comparado com papéis do ATT e do GATT.

O GAP estabelece quatro funções com requisitos específicos no controlador subjacente: função emissora, observadora, periférica e central. No caso de um dispositivo na função *broadcaster* este somente transmite dados (através dos canais de publicidade) e não suporta conexões com outros dispositivos. Sendo assim, possui o papel de observador é complementar para o *broadcaster*, ou seja, tem o objetivo de receber os dados transmitidos pelo recetor de ondas rádio. A função Central é projetada para um dispositivo encarregado de iniciar e gerir várias conexões, enquanto a função Periférica é projetada para dispositivos simples que usam uma única conexão com um dispositivo na função Central. Consequentemente, as funções Central e Periférica exigem que o controlador do dispositivo suporte as funções mestre e escravo, respetivamente. Um dispositivo pode suportar várias funções, mas apenas uma função pode ser adotada

num determinado mo- mento. Para além destes perfis podem ser criados outros perfis adicionais sobre a camada GAP. O Bluetooth segue uma hierarquia de perfis, na qual um novo perfil, incluindo todos os requisitos de um perfil existente, podem ser definidos. Um perfil de alto nível é chamado de perfil de aplicação e especifica como as aplicações podem interoperar. Os perfis de aplicação, além de favorecerem a interoperabilidade entre as aplicações também favorecem a interoperabilidade entre dispositivos de origens distintas, ou seja, de diferentes fabricantes[12].

Na Figura 19 é apresentado o diagrama de estados referente à camada GAP. Esta figura pretende identificar os diferentes estados que o dispositivo toma ao longo dos processos que acontecem na camada GAP: "Em espera", "Anunciante", "Scanner", "Iniciador", "Mestre" e "Escravo".

- Em espera: primeiro estado do dispositivo;
- Anunciante: o dispositivo anuncia com dados específicos, o que permite que qual- quer dispositivo iniciante consiga identificá-lo como um dispositivo conectável;
- Scanner: ao receber o anúncio, o dispositivo que realiza o *scan* envia uma solicitação ao anunciante. O anunciante responde com uma resposta de verificação. Todo este processo é chamado de descoberta de dispositivo. Neste processo é também iniciada a conexão entre o dispositivo de *scan* e o de publicidade;
- Iniciador: ao iniciar, o iniciador deve especificar um endereço de dispositivo ao qual se conectar. A conexão deve ser estabelecida consoante alguns parâmetros: intervalo de conexão, latência do escravo, *time-out* de supervisão;
- Escravo/mestre: quando uma conexão é formada, o dispositivo funciona como escravo se o anunciante e o mestre funcionarem como iniciador.

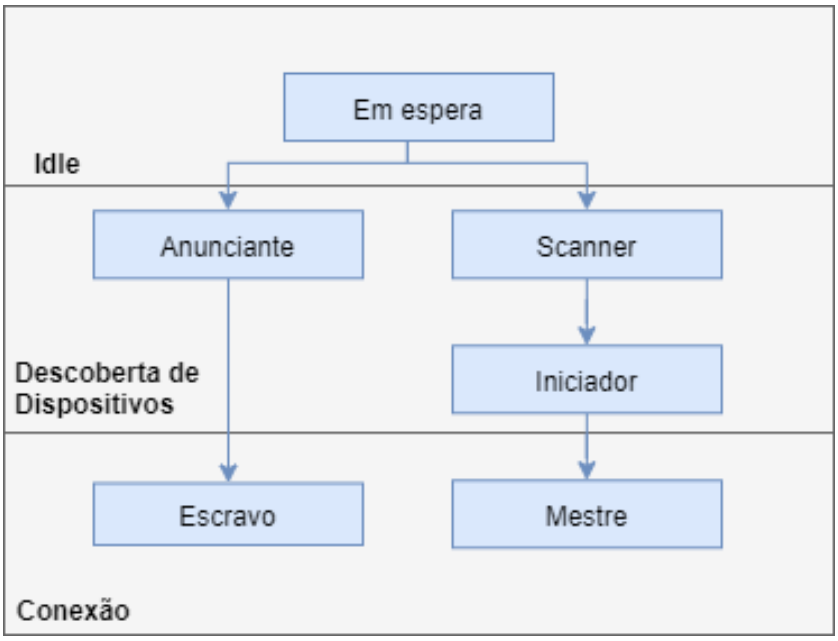

Figura 19 Diagrama de Estado do GAP (Adaptado[11])

Os parâmetros de conexão, já referidos, são: intervalo de conexão, latência do escravo, time-out do escravo.

Nas comunicações BLE é utilizado o esquema de *frequency-hopping*. Os dispositivos de anúncio e o de iniciação enviam e recebem dados um do outro apenas a partir de um canal específico num determinado intervalo de tempo. Esses dispositivos voltam a comunicar-se após um período específico de tempo num novo canal. Este encontro entre os dois dispositivos em que estes enviam e recebem dados é conhecido como um evento de conexão, representado na Figura 20. Se não houver dados a serem enviados ou recebidos, os dois dispositivos comunicam (enviam os dados) através da camada de ligação para que a conexão seja mantida. O parâmetro chamado de intervalo de conexão, também presente na Figura 20 quantifica o tempo de conexão entre dois eventos em unidades de 1,25 ms. Este intervalo de conexão pode variar entre um valor mínimo de 6 (7,5 ms) e um máximo de 3200 (4,0 s).

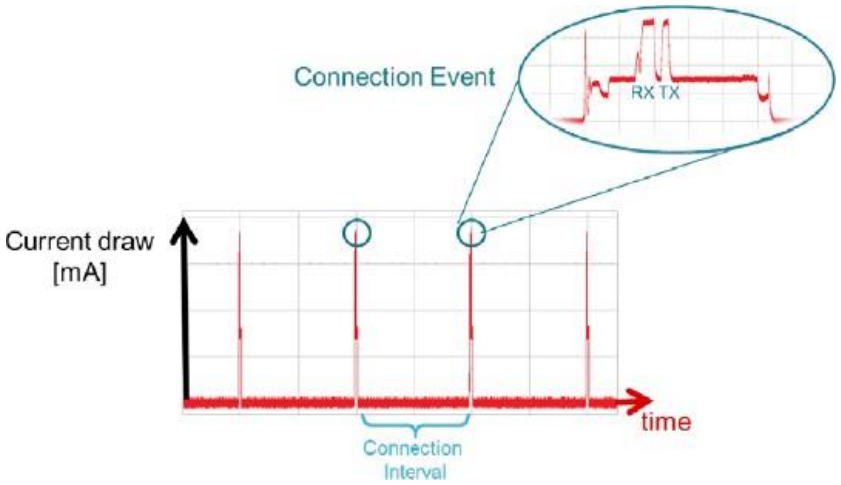

Figura 20 Evento e intervalo de conexão[11]

A latência do escravo é um parâmetro que concede ao dispositivo escravo (periférico) a opção de ignorar vários eventos de conexão. Esta capacidade fornece ao dispositivo escravo uma maior flexibilidade. Se não existirem dados para enviar o escravo, poderá "saltar" eventos de conexão, permanecer em suspensão e poupar energia. Apesar de o escravo poder ignorar eventos de conexão, não deve ignorar mais do que o permitido pelo parâmetro de latência do escravo pois, caso esse limite seja ultrapassado, a conexão poderá falhar.

O time-out (tempo limite) da supervisão trata-se do intervalo de tempo máximo entre dois eventos de conexão. Se esse tempo for alcançado sem que haja qualquer evento de conexão bem-sucedido, o dispositivo terminará a conexão e retornará a um estado de suspensão ou conexão. Este parâmetro é representado em unidades de 10 ms, variando entre um mínimo de 10 (100 ms) até 3200 (32,0 s). O tempo limite deve ser superior ao intervalo de conexão efetivo[11].

2.4 Sistemas Operativos em tempo real para Sistemas Embebidos: caso do TI-RTOS

Um sistema operativo em tempo real, do inglês *Real Time Operating System* (RTOS) é um software de sistema que fornece serviços e gere recursos de processador para aplicações. Este sistema inclui recursos, tais como ciclos de processador, memória, periféricos e interrupções. O Objetivo da gestão dos ciclos do processador é fazer o escalonamento das tarefas, isto é "repartir"o tempo de processamento entre várias tarefas que o software incorporado deve executar. Por isso, o sistema operativo em tempo real controla a execução de *thread* e gere o contexto de cada *thread*. Cada *thread* recebe uma prioridade pelo designer, para controlar qual a *thread* a executar, em caso de concorrência pelos recursos. Quando uma *thread* de prioridade mais alta necessita de ser executada, o RTOS guarda o contexto das *threads* atualmente em

execução na memória e restaura o contexto da nova *thread*, designando-se esta atividade por comutação de contexto.

2.4.1 TI-RTOS

O TI-RTOS é um sistema operativo para projetos onde pode ser usado o BLE, e deriva do ambiente operacional RTOS. O *kernel*, ou núcleo, do TI-RTOS é uma versão personalizada do *legacy kernel* SYS/BIOS. Este TI-RTOS opera da mesma forma que um sistema operativo *multithread* que opera preemptivamente e em tempo real e que possui drivers, ferramentas de sincronização e *scheduling*.

Para além do conceito de prioridade associado a cada tarefa, o TI-ROS, como tipicamente outros RTOSs, executa código em diferentes contextos, como apresentado na Figura 21. Quando há uma interrupção de HW (ativada por um evento externo), é executado o código de atendimento à interrupção. O processador suspende por hardware a execução das tarefas e este código é executado até ao fim. Um outro contexto de interrupção é o das interrupções por software (tipicamente ativadas dentro das interrupções de hardware) que têm uma prioridade inferior às de hardware. Ou seja, uma interrupção de Sw só pode ser interrompida por uma de HW, ou por outra de SW mas de maior prioridade, sendo a comutação do seu contexto gerida pelo TI-RTOS. Fora do contexto de interrupções, isto é, o modo normal de execução do processador, executam as tarefas. Estas são diferenciadas pelo TI-ROS em função da sua prioridade, especificada aquando do design da solução. Quando nenhuma tarefa está a ser executada, o TI-RTOS põe em execução a tarefa Idle, uma tarefa predefinida com prioridade inferior a todas as outras tarefas do sistema.

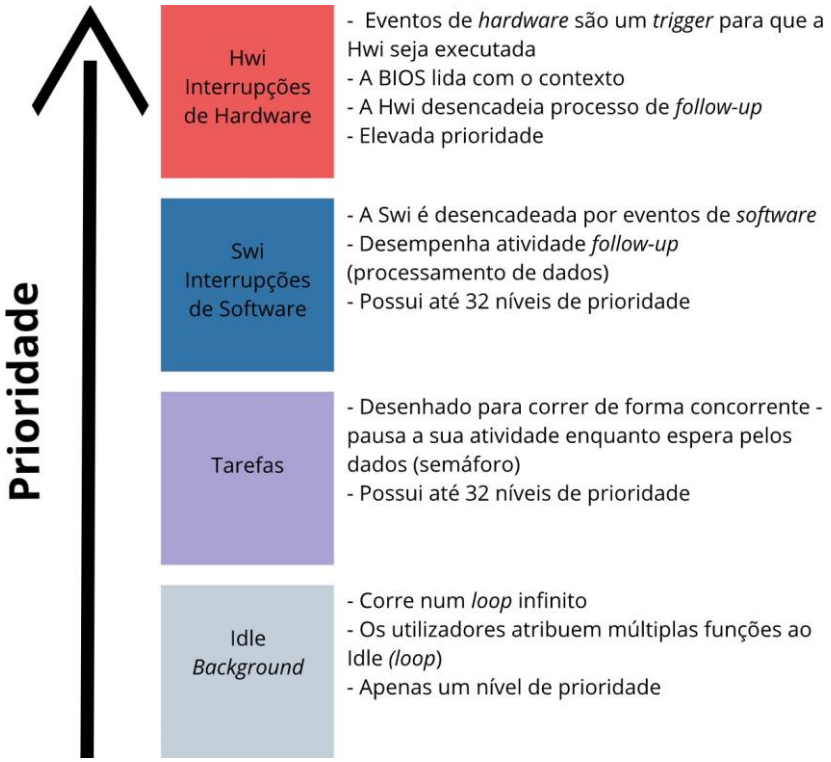

Figura 21 Execução das Threads TI-RTOS (Adaptado [11])

2.4.2 Interrupções de hardware (Hwi)

As Hwi (também chamados de rotinas de serviço de interrupção ou RSI) são as *threads* com a maior prioridade no contexto de uma aplicação TI-RTOS. Estas *threads* são usadas para executar tarefas críticas que possuem prazos críticos. Estas *threads* são desencadeadas como resposta a eventos assíncronos externos (interrupções) que ocorrem em tempo real. No caso das *threads* da Hwi são sempre executados até à sua conclusão, no entanto podem ser impedidas temporariamente pelas *threads* do Hwi desencadeadas por outras interrupções.

Geralmente, as RSIs são de curta duração para que os requisitos de tempo real do sistema não sejam afetados. Para além disso, nenhum evento que cause uma interrupção pode ser chamado dentro desse contexto, já que as Hwis devem ser executadas até à sua conclusão.

No caso específico do CC2640R2F[], o módulo Hwi também suporta interrupções de latência zero, isto é, interrupções que não geram atrasos já que as Hwis são eventos de maior prioridade. Essas interrupções não necessitam de passar pelo *dispatcher* TI-RTOS Hwi e, portanto, são mais responsivas e eficientes que as interrupções ditas normais, ou padrão. Apesar de tudo, o *dispatcher* impede que a RSI invoque diretamente as APIs do *kernel* TI-RTOS. Cabe à rotina de serviços de interrupção preservar o seu próprio contexto de forma a impedir que ele interfira no *scheduler* do *kernel*[15].

2.4.3 Interrupções de software (Swi)

As interrupções de software (Swi) são geradas após as interrupções de hardware (Hwi), as *threads* de interrupção de software fornecem níveis de prioridade intermédios. Ao contrário da Hwis, desencadeadas por interrupções de hardware, os Swis são desencadeadas de forma programada, chamando algumas das APIs do módulo Swi.

Tal como as *threads* de interrupções de hardware, as *threads* de interrupções de software são sempre executadas até ao fim e não podem ser bloqueadas, por exemplo por um semáforo.

Tal como as Hwis, as interrupções de software devem ser curtas e não incluir chamadas de eventos que causem interrupções. Isso permite que tarefas de alta prioridade sejam executadas em conformidade.

2.4.4 Tarefas

As *threads* de Tarefas executam no contexto de execução normal do TI-RTOS e podem distinguir-se entre si por diferentes níveis de prioridade, todos superiores à da tarefa de background (Idle) e menor que as interrupções de software. As tarefas diferem das *interrupções* de software, pois conseguem esperar, durante a execução de tarefas de prioridade superior, que os recursos estejam disponíveis. Por outro lado, as tarefas exigem uma *stack* separada para cada *thread*. O TI-RTOS fornece vários mecanismos que podem ser usados para comunicação e sincronização entre tarefas, incluindo semáforos, eventos, filas de mensagens e caixas de correio.

A tarefa Idle corresponde à tarefa com a prioridade mais baixa no contexto de uma aplicação TI-RTOS. A aplicação TI-RTOS chama a rotina de inicialização de cada módulo TI-RTOS e, em seguida, entra na tarefa. Tipicamente esta tarefa não executa qualquer *thread*, no entanto a existir, é o código de menor prioridade do sistema, executando apenas se mais nenhuma *thread* estiver a executar[15].

2.4.5 ICall

O ICall (Indirect Call Framework) é um módulo que fornece a capacidade da aplicação interagir com os serviços da *stack* de protocolos BLE, bem como com determinados serviços primitivos fornecidos pelo RTOS. A função ICall permite que a pilha de aplicações e protocolos aja com eficiência, consiga comunicar e compartilhar recursos num ambiente unificado em tempo real.

A componente central da arquitetura ICall é o elemento *dispatcher*, o que simplifica a interface do programa de aplicações entre a aplicação. Embora a maioria das interações da ICall sejam abstraídas nas APIs da *stack* de protocolos BLE, o *developer* da aplicação deve assimilar a arquitetura subjacente para a *stack* de protocolos operar adequadamente no ambiente RTOS *multithread*.

Como mostra a Figura 22, a principal razão para o uso da ICall envolve o envio de mensagens entre uma entidade-servidor e uma entidade-cliente.

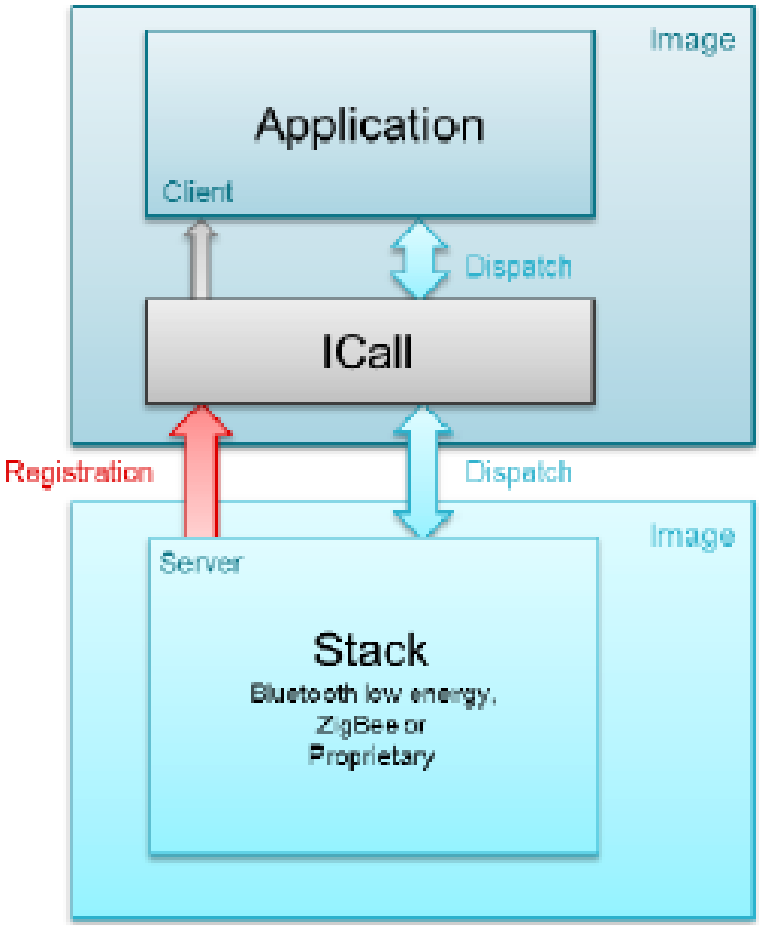

Figura 22 Arquitetura Icall [11]

O serviço da *stack* de protocolos BLE ICall serve de interface para as APIs de pilha da BLE. Quando uma API da pilha é chamada por uma aplicação, o módulo ICall redireciona o comando, de novo, para a pilha e envia a mensagens para a aplicação quando for adequado.

Como a ICall faz parte do projeto da aplicação, a ICall pode ser acedida através de chamadas de função diretas. A tarefa da aplicação é bloqueada até que a resposta seja recebida já que a pilha BLE é executada com maior prioridade.

Algumas APIs da stack podem responder de forma assíncrona à aplicação, por meio do ICall, devido à resposta enviada ao manipulador de eventos da tarefa da App.

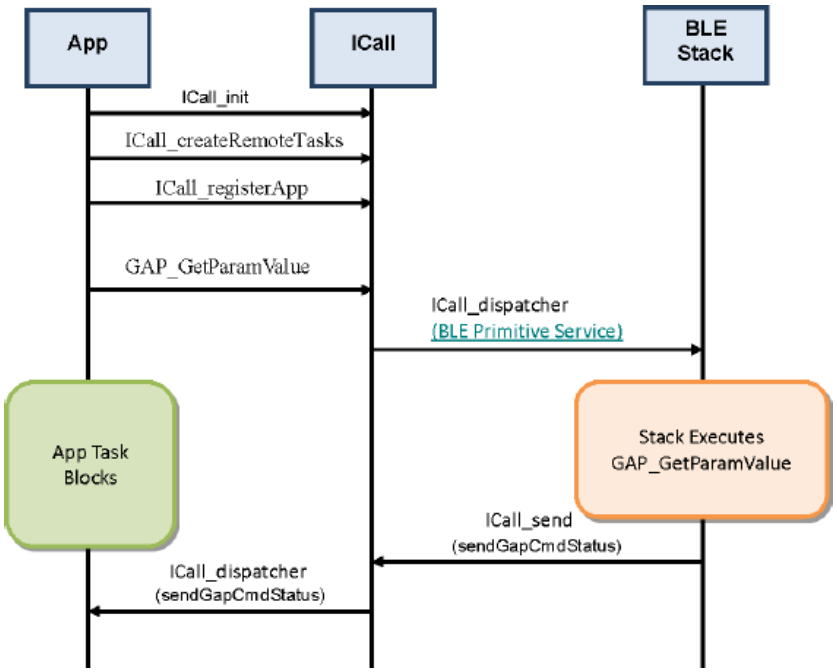

Figura 23 Exemplo de uso da ICALL [11]

A Figura 23 representa o diagrama de sequência de um exemplo de comando que está a ser enviado da aplicação para a *stack* de protocolos BLE através da estrutura ICall com um valor de retorno correspondente ao devolvido para a aplicação.

ICall_init () inicializa a instância do módulo ICall, enquanto que o método ICall_createRemoteTasks () cria uma tarefa por imagem externa com uma função de entrada associada a um endereço conhecido. Após inicializar a ICall, a tarefa da aplicação é registada no ICall através do método ICall_registerApp (). De seguida, o *scheduler* SYS/BIOS ser inicializado e a tarefa da aplicação ser executada, a aplicação envia um co- mando definido no ICallBLEAPI.c, como GAP_GetParamValue (). O comando *protocol* não é executado no encadeamento da aplicação, mas é encapsulado numa mensagem ICall e reencaminhado para a tarefa de pilha BLE através da ICall. Este comando é enviado ao distribuidor ICall, onde é expedido e executado no lado do servidor. Enquanto isso, o encadeamento da aplicação bloqueia a mensagem de status do comando correspondente. Quando a pilha termina de executar o comando, a resposta da mensagem de status é enviada pelo ICall de volta ao encadeamento da aplicação.

2.5 Microcontrolador CC2640R2F

Para a aquisição de dados provenientes do sensor foi utilizado uma placa de desenvolvimento da Texas Instruments, LAUNCHXL-CC2640R2F apresentada na Figura 24, que também é utilizada para a transmissão de dados via bluetooth para um dispositivo externo (telemóvel, computador ou tablet).

O dispositivo CC2640R2F é um microcontrolador (MCU) sem fios destinado a aplicações Bluetooth Low Energy 4.2 e 5. O dispositivo é um membro da família CC26xx SimpleLink de potência ultrabaixa, dispositivos de Radio-Frequência (RF) de 2,4 GHz de baixo custo. O consumo de corrente ativa de RF muito baixa e MCU, em modo corrente de baixo consumo, fornecem um excelente tempo de vida útil à bateria o que permite a operação em pequenas baterias de célula e em aplicações de recolha de energia.

O dispositivo CC2640R2F contém um núcleo ARM Cortex-M3 de 32 bits que é executado a 48 MHz como processador principal e um rico conjunto de recursos periféricos que inclui um controlador, exclusivo para sensores, de potência ultrabaixa. Este controlador é ideal para interface de sensores externos e para recolher dados analógicos e digitais de forma autónoma, enquanto que o restante sistema está em modo de suspensão. Portanto, o dispositivo CC2640R2F é compatível com uma ampla gama de aplicações em que é bastante importante a redução do consumo de energia permitindo um tempo elevado de vida útil da bateria, o seu formato pequeno e a facilidade de uso. Os sistemas de rádio e de gestão de energia e, o relógio do MCU sem fio CC2640R2F requerem uma configuração e utilização específicas por software para operar corretamente, tal como o que foi implementado no TI-RTOS. O controlador do Bluetooth Low Energy e as bibliotecas de *host* são incorporados na ROM (Read Only Memory) e executados, de forma parcial, num processador ARM Cortex -M0. Esta arquitetura melhora o desempenho do sistema, de forma generalizada, o consumo de energia e liberta quantidades significativas de memória flash para a aplicação[13].

Figura 24 LAUNCHXL-CC2640R2F

Na Figura 25 encontra-se representado o diagrama de bloco do dispositivo CC2640R2F onde são apresentados as suas principais caraterísticas.

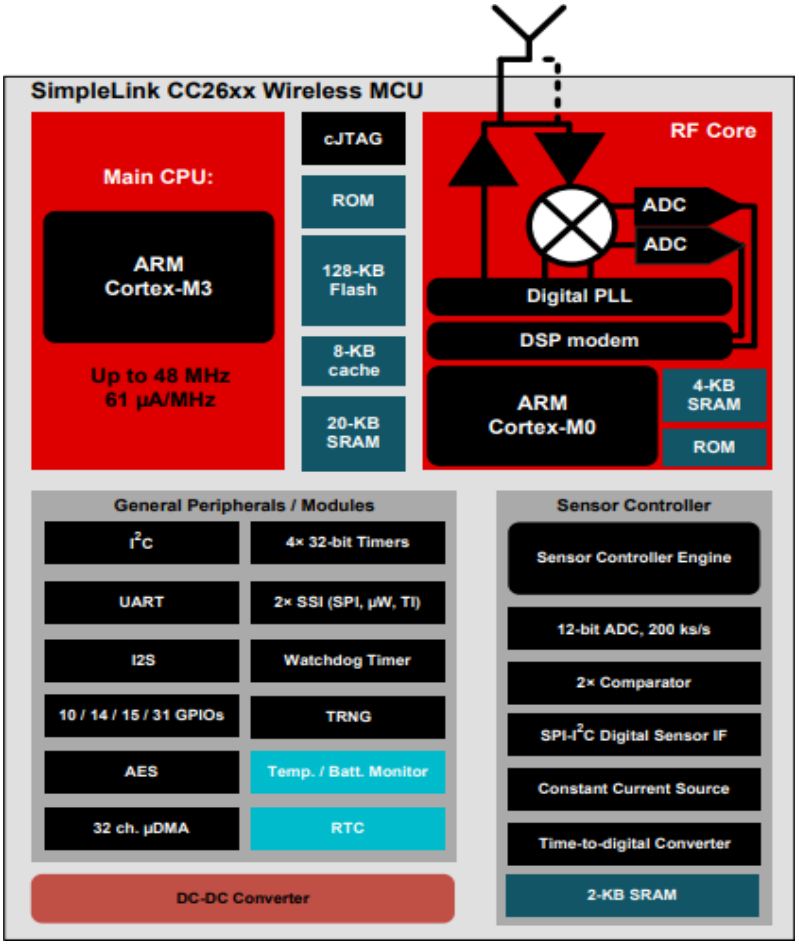

Figura 25 Diagrama Funcional-CC2640R2F[13]

O núcleo do Cortex M0 (CM0) no CC2640R2F é responsável pela interface com o *hardware* do rádio e pela tradução de instruções complexas do núcleo do Cortex M3 (CM3) em bits enviados pelo ar usando o rádio. Para o protocolo BLE, o CM0 implementa a camada PHY da pilha de protocolos. Frequentemente, o CM0 é capaz de operar de forma autónoma, o que possibilita o CM3 para processamento de protocolo e camada da aplicação de nível superior.

O núcleo do sistema (CM3) é projetado para executar a pilha de protocolos BLE da camada de *link* até a aplicação do utilizador. A camada de *link* faz interface com o núcleo do rádio por meio de um módulo de *software* chamado *driver* de RF, que fica acima da campainha da RF. O *driver* de RF é executado no CM3 e atua como uma interface para o rádio no CC2640R2F e também gere os domínios de energia do *hardware* e do núcleo do rádio.

Capítulo 3 – Análise do Problema

Neste capítulo é abordada e analisada a definição do problema e as suas motivações, são apresentados os seus requisitos, caso de uso, e também é apresentada a arquitetura do sistema desenvolvido.

3.1 Definição do Problema

Pretende-se com este projeto o desenvolvimento de um sistema de aquisição de sinal sem fios para aplicação no desporto com o objetivo de analisar a variação de sinal que se encontra associada às forças de impacto sobre o sensor, tendo como caso de uso a medição da força exercida e o tempo de reação.

O sistema implementado é composto pela placa CC2640R2F da Texas Instruments, principal módulo, e pela aplicação Android, que serve de apoio. A placa desempenha o papel de servidor recebendo os sinais provenientes do sensor e realizando a conversão ADC. Numa fase seguinte, os dados são enviados para a aplicação Android através do mecanismo de comunicação BLE. Após serem recebidos, os dados passam a estar disponíveis na aplicação, onde é possível retirar algumas conclusões. Apesar de estar aplicado, nesta dissertação ao âmbito da prática do boxe, este sistema de aquisição pode ser utilizado em outras áreas de aplicação que possam ser auxiliadas por ferramentas de monitorização como esta.

3.2 Requisitos

O sistema desenvolvido foi concebido tendo em conta os objetivos, funcionalidades e restrições de software e hardware. Sendo assim, foi necessário desenvolver uma lista de requisitos (funcionais e não funcionais) que representassem essas condições, abaixo descritas.

Requisitos Funcionais

- O sistema deve adquirir, armazenar e visualizar dados inerentes à força aplicada ao sensor;
- O sistema deverá medir o tempo da resposta do atleta depois de sinalizada a ordem de ação (acionamento de um *led* no sistema de aquisição e condicionamento do sinal);

Requisitos Não Funcionais

- Comunicação sem fios para a plataforma de processamento
- Plataforma de receção dos dados baseada em telemóvel
- Configuração do número de canais a adquirir
- Configuração das frequências de amostragem

3.3 Caso de uso

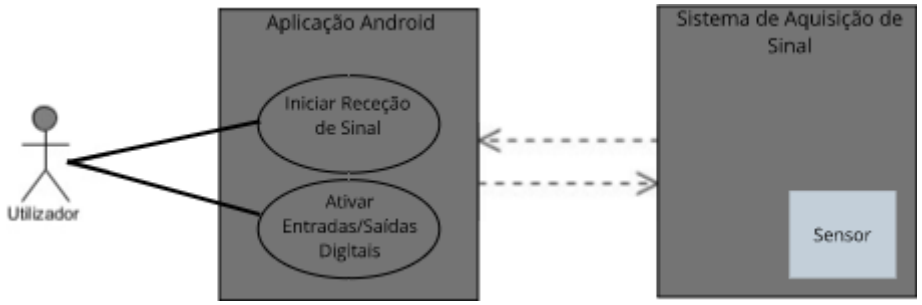

Figura 26 Caso de Uso

O caso de uso apresentado na Figura 26 representa as funcionalidades da aplicação Android a que o utilizador tem acesso. O utilizador pode selecionar entre a opção "iniciar receção de sinal" e "ativar entradas/saídas digitais".

Para além da interação que possui com o utilizador, a aplicação Android também interage com o sistema de aquisição de sinal, o qual é responsável pela receção e envio de informação, que depois passa a estar disponível para o utilizador.

3.4 Diagramas de Sequência

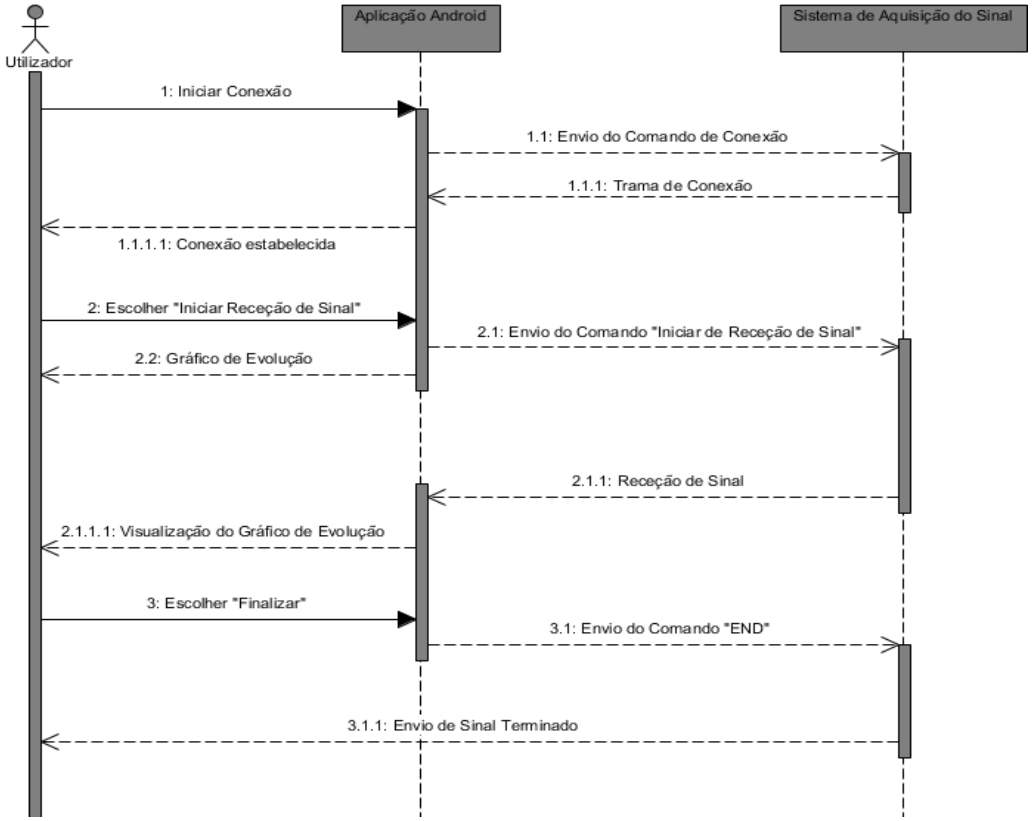

Figura 27 Diagrama de sequência da receção de sinal

A Figura 27 representa o diagrama de sequência para a funcionalidade "Iniciar Receção de Sinal".

Para o utilizador aceder a esta funcionalidade deve estabelecer conexão com a aplicação e, a partir desse momento é capaz de fazer uso da funcionalidade. A funcionalidade é caracterizada por um conjunto de fases. Numa primeira instância, após o utilizador selecionar a funcionalidade na aplicação é enviado um comando de "iniciar receção de sinal" para o sistema de aquisição de sinal, de forma automática. De seguida, e quase logo após o envio do comando para o sistema de aquisição de sinal, o utilizador recebe como resposta da aplicação Android um gráfico, o gráfico de evolução. Este gráfico é responsável por revelar as variações de sinal que vão acontecendo no sistema de aquisição de sinal.

Após a "receção de sinal" e a "visualização do gráfico de evolução", o utilizador pode escolher a opção "finalizar" e assim interromper o envio de sinal, através do envio do comando "end" para o sistema de aquisição de sinal.

Após o comando ser recebido pelo sistema de aquisição de sinal a comunicação entre a aplicação Android e o sistema de aquisição de sinal termina.

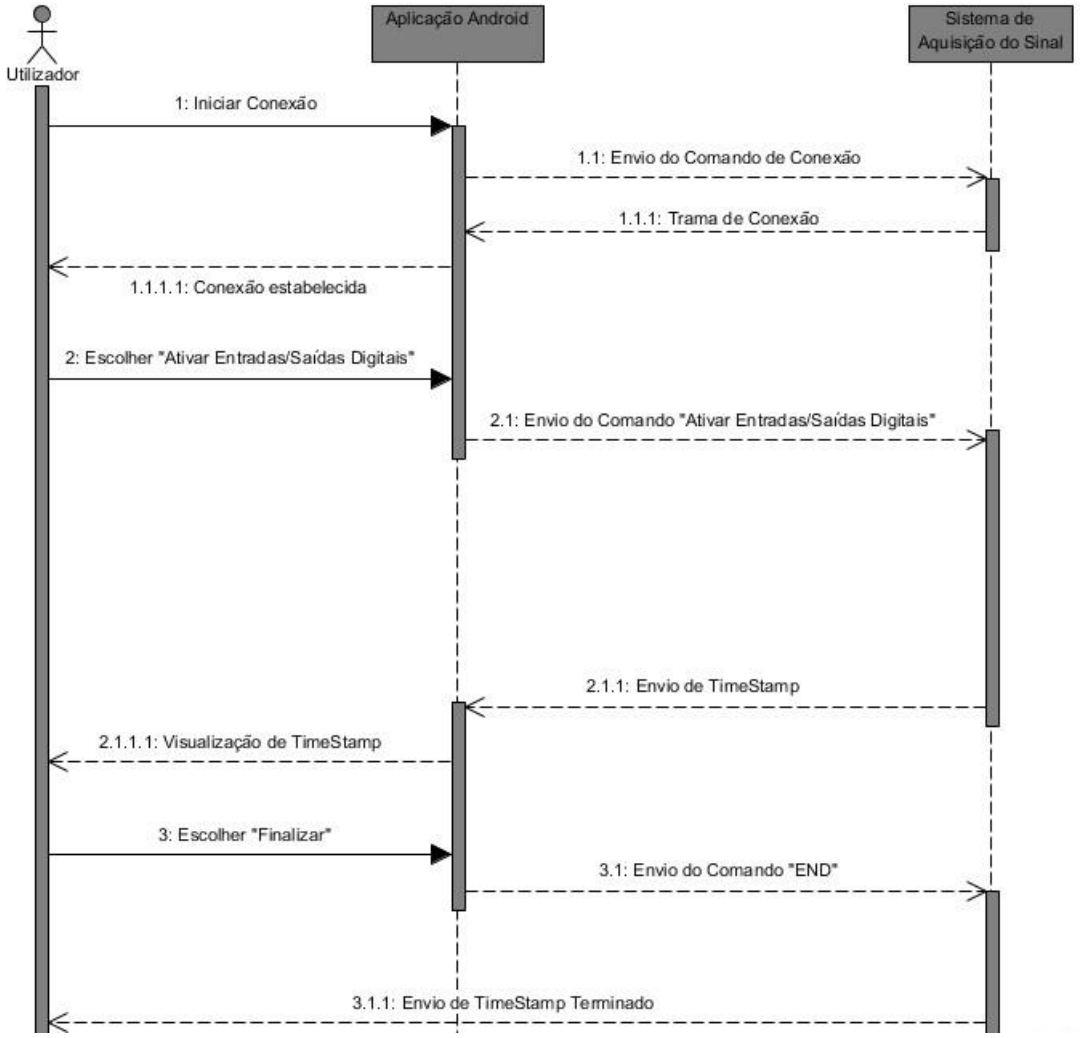

Figura 28 Diagrama de sequência Ativar Entradas/Saídas digitais

A Figura 28 representa o diagrama de sequência para a funcionalidade ”ativar entradas/saídas digitais”.

Para o utilizador aceder a esta funcionalidade deve estabelecer conexão com a aplicação e, a partir desse momento é capaz de fazer uso da funcionalidade. A funcionalidade é caracterizada por um conjunto de fases. Numa primeira instância, após o utilizador selecionar a funcionalidade na aplicação é enviado, automaticamente, um comando de ativação das entradas/saídas digitais da aplicação Android para o sistema de aquisição de sinal. De seguida, o sistema de aquisição de sinal inicia o ”envio de timestamp”, a qual é enviada para utilizador através da aplicação Android, ”visualização de timeStamp”.

Após o ”envio de timestamp” e a ”visualização de timestamp”, o utilizador pode escolher a opção ”finalizar” e assim interromper o envio de timestamp, através do envio do comando ”end” para o sistema de squisição de sinal, através da aplicação Android. Numa fase final, a comunicação entre a aplicação Android e o sistema de aquisição de sinal termina com a mensagem envio de “timestamp terminado” que permite calcular o tempo de reação.

3.5 Arquitetura do Sistema

A arquitetura do sistema apresentada na Figura 29 refere-se a um sistema de aquisição pelo que apresenta:

- Sensor
- Condicionamento
- Aquisição
- Envio (BLE)
- Processamento

Todos esses componentes são abordados de uma forma mais detalhada no capítulo que se segue, o qual descreve os componentes individualmente explorando o seu papel e funcionalidades no contexto deste projeto.

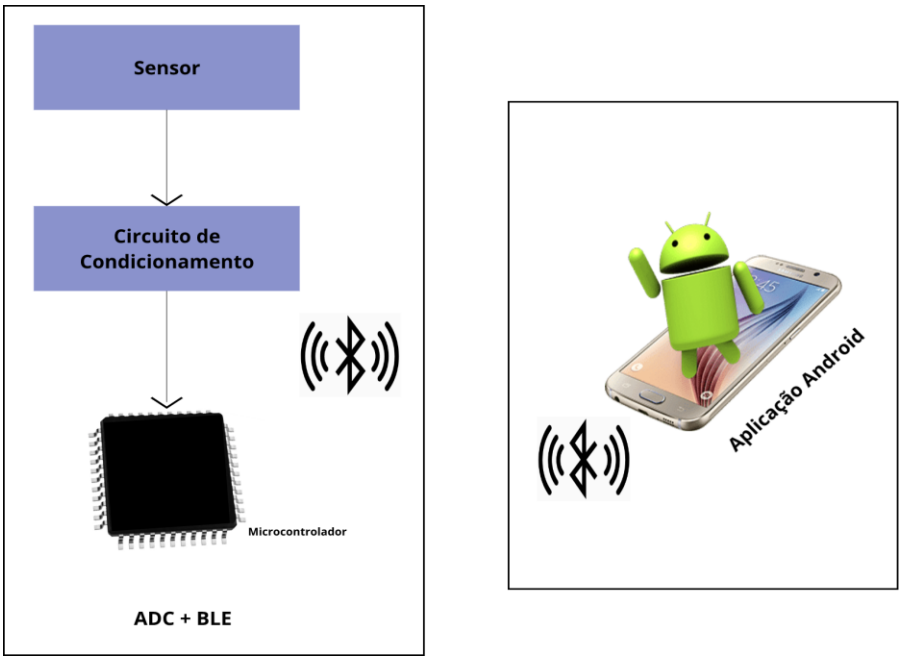

Figura 29 Arquitetura geral do sistema

Capítulo 4 – Especificação e Implementação do Sistema

Neste capítulo são apresentadas as principais partes do sistema desenvolvido, apre- sentado na Figura 29, assim como os seus componentes, referidos no final do capítulo anterior: a interação entre a aplicação Android e o sistema de aquisição de sinal.

4.1 Especificação da Solução

Como referido na secção da arquitetura do sistema no capítulo anterior, o bloco que se encontra na Figura 30 refere-se ao sistema de aquisição e condicionamento de sinal que é composto pelo sensor, um circuito de condicionamento e um microcontrolador (CC2640R2F)[13] responsável pela conversão do sinal ADC e pelo envio de dados via Bluetooth para a aplicação Android.

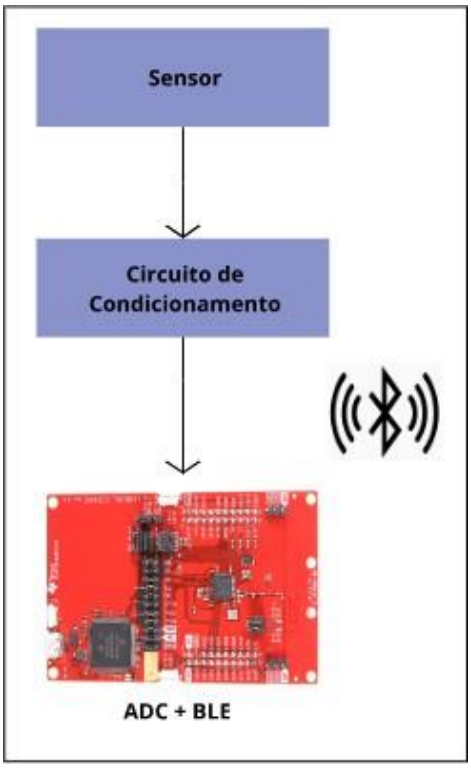

Figura 30 Sistema de aquisição e condicionamento do sinal

4.1.1 Especificação da Solução

O *Software* de aquisição de sinal é constituído por um serviço implementado na estação periférica tomando esta o papel de GATT Server (Servidor) formado por duas características, uma responsável por envio periódicos de dados por notificação, denominado ”DataOut”cujo valor é uma trama de 20 bytes e um descritor (CCCD) com o intuito de ativar o modo de envio por notificações (0x0001), e uma outra característica ”DataIn” responsável pela receção de dados proveniente do cliente que tem o papel de GATT

Client (Cliente). As notificações, ao invés das indicações, são úteis para este caso, porque permitem o envio de dados do dispositivo periférico para o central sem que este os solicite e envie a confirmação de receção (ACK), pelo que o *overhead* é menor, tornando o processo de transmissão de dados mais eficiente. Na Figura 31 é possível observar a constituição deste serviço.

Figura 31 Características do serviço Simple Serial Socket Server

Para uma melhor compreensão do *software* de aquisição desenvolvida são apresentados nas Figuras 32, 33 e 34 os fluxogramas das partes consideradas mais importantes.

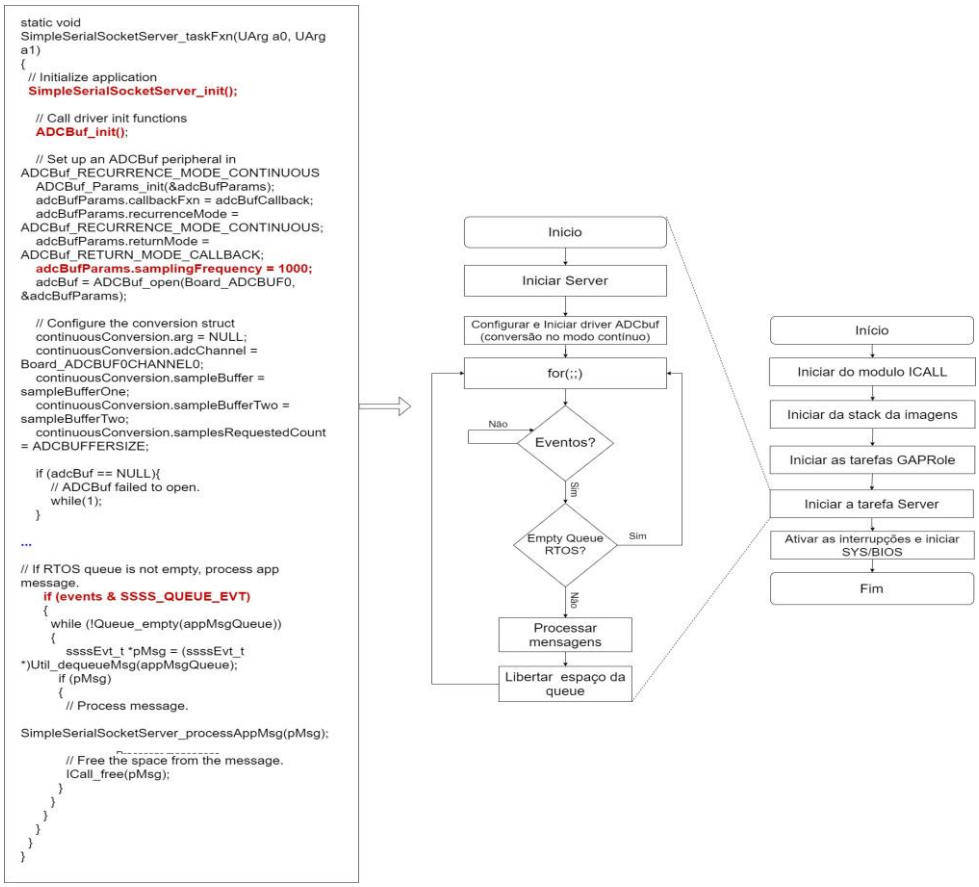

Figura 32 Fluxograma da aplicação

No caso da Figura 32 , pode destacar-se a tarefa cujo nome é "Iniciar tarefa Server". Esta tarefa é o núcleo da aplicação. Na Figura do lado esquerdo é possível visualizar código usado para sua construção. As funções a vermelho são responsáveis pela inicialização e configuração do hardware, inicialização e especificação dos parâmetros do driver AdcBuffer e verificação da existência de dados na *queue* RTOS. Do lado direito a tarefa "Iniciar tarefa Server" encontra-se representado o fluxograma que mostra a tarefa como um todo, isto é, como um conjunto de eventos que possuem alguma complexidade.

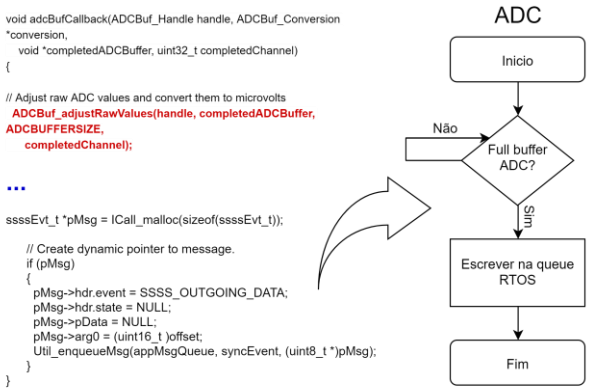

Figura 33 Fluxograma conversões ADC

No contexto da tarefa anteriormente referida, ”Iniciar tarefa Server”, depois da inicialização e especificação dos parâmetros do driver AdcBuffer, é escrita na *queue* RTOS os valores das conversões ADC sempre que o buffer ADC estiver cheio. Na Figura 33 encontra apresentado um trecho de código e o respetivo fluxograma da função (adcBufCall_back(ADCBuf_Handle handle, ADCBuf_Conversion *conversion, void *completedADC_Buffer, uint32_t completedChannel)) que é invocada sempre que o buffer ADC estiver cheio, onde é possível encontrar a função ”ADCBuf_adjustRawValues” responsável por lidar com a conversão dos valores ADC apresentadas.

```
static void SimpleSerialSocketServer_processAppMsg(ssssEvt_t *pMsg)
{
  switch (pMsg->hdr.event)
  {
   // Outgoing data event
   case SSSS_OUTGOING_DATA:
     {
       // You could do processing of data here ...
       // For now, just Send the data read from Queue RTOS over the air
        bStatus_t status = SimpleStreamServer_sendData(connHandle,
&adcTxBuffer, pMsg->arg0);

       // If status is not successful, register for connection events for further
processing
       if (status != SUCCESS) {

SimpleSerialSocketServer_RegisterToAllConnectionEvent(FOR_STREAM);
       }
       break;
     }
```

Figura 34 Fluxograma processamento de mensagens

Ainda no mesmo contexto da Figura anterior, é apresentada a Figura 34 que visa fornecer um maior destaque à fase “processar mensagens” desta mesma tarefa. Do seu lado esquerdo encontra-se destacada a função “SimpleStreamServer_sendData” responsável pelo envio de dados via Bluetooth.

4.1.2 Receção e Processamento do Sinal

Como referido na secção da arquitetura do sistema no capítulo anterior, este bloco que se encontra na Figura 35 refere-se ao cliente(aplicação Android) , sistema responsável pela receção, processamento e envio de comandos (Start/Stop e Acionar).

Figura 35 Sistema de aquisição e condicionamento do sinal

Para o desenvolvimento da aplicação Android foram estudados e implementados módulos de código, como por exemplo a biblioteca Blessed, que contribuiriam para a implementação/cumprimento dos requisitos do problema.

A aplicação Android desenvolvida no âmbito desta dissertação é a "SportsInfo". Esta aplicação estabelece a conexão entre o sistema de aquisição de sinal e o dispositivo Android, de forma a que os dados possam ser rececionados e consultados num dispositivo móvel. No contexto do sistema, a aplicação Android é o cliente e o sistema de aquisição de sinal é o servidor, já que a primeira entidade recebe a informação da segunda entidade.

Aquando da conexão, o cliente (aplicação Android) ativa uma notificação de forma a estar disponível para receber os dados do servidor (sistema de aquisição de sinal).

Para além disso, a aplicação é capaz de enviar comandos de "Start and Stop" e permite ativar leitura/escrita de entradas e saídas digitais.

As Figuras 36 e 37 correspondem sucessivamente a códigos implementados para a ativação de notificações e a receção dos dados enviados do sistema de aquisição de sinal.

Figura 36 Ativação da notificação

Para receber os dados do servidor, o cliente tem de ativar a notificação ou a indicação. A escolha da opção notificação, ao invés da indicação, deve-se ao fato das indicações necessitarem de ser confirmadas pela pilha Bluetooth o que faz com que haja um pequeno desperdício de tempo, enquanto que as notificações não precisam de ser confirmadas.

Para ativar as notificações, como apresentado na Figura 36 setCharacteristicNotification faz com que a pilha Bluetooth espere notificações para a característica f000c0c2-0451- 4000-b000-000000000000 escrevendo 1 como um int16 (0x0001) no *Client Characteristic Configuration Descriptor* (CCCD).

Depois de ativar a notificação sempre que houver atualizações de dados o periférico (servidor) envia os dados para a central (cliente).

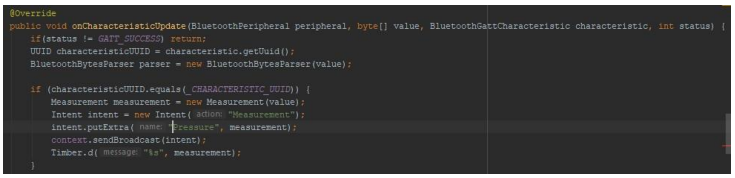

Figura 37 Códigos da Receção dos dados

4.2 Implementação

4.2.1 Aquisição e Envio – Placa TI

Para a implementação do Bluetooth no kit de desenvolvimento Launchpad CC2640R2F, foram utilizados dois exemplos do software fornecido pela Texas Instruments, Simple_Serial_Socket_Server e Adcbufcontinuous.

O propósito do exemplo Simple_Serial_Socket_Server é demonstrar um serial *socket* a partir do uso do serviço *Simple Stream Server* da TI. O serviço foi desenvolvido para implementar uma conexão de fluxo de dados bidirecional sobre o protocolo BLE, usando um UUID de 128 bits: F000C0C0-0451-4000-B000-00000000-0000 e contém duas caraterísticas (*DataIn* e *DataOut*) como foi referido na seção de aquisição e envio do sinal. O exemplo mostra uma implementação simples de UART sobre BLE. Sendo projetado para permitir que o real coletor de dados e a origem sejam facilmente trocados com a

finalidade de criar uma aplicação de fluxo simples genérico.

O Adcbufcontinuous é um exemplo que usa o driver ADCBuf no modo contínuo para preencher um buffer de amostra do tamanho 100 e, em seguida, usa o DMA para transferir os resultados para a SRAM,

que depois é enviada via UART, através do micro- controlador. Os valores exibidos são conversões ADC em micro-volts como pode ser visto no exemplo da Figura 38. Uma única tarefa configura a conexão UART da porta série e inicializa a conversão contínua com o driver ADCBuf que gera 200 amostras a 200 Hz e, após estas operações a tarefa entra num ciclo infinito. O driver ADCBuf suporta taxas de amostragem mais altas; no entanto, 200 Hz foi a frequência escolhida para uma saída facilmente interpretável.

Este exemplo possui uma *Interrupt Service Routine* (ISR) **adcBufCallback** que é invocada sempre que um buffer ADC estiver cheio. O conteúdo do buffer é convertido em formato legível e enviado via UART.

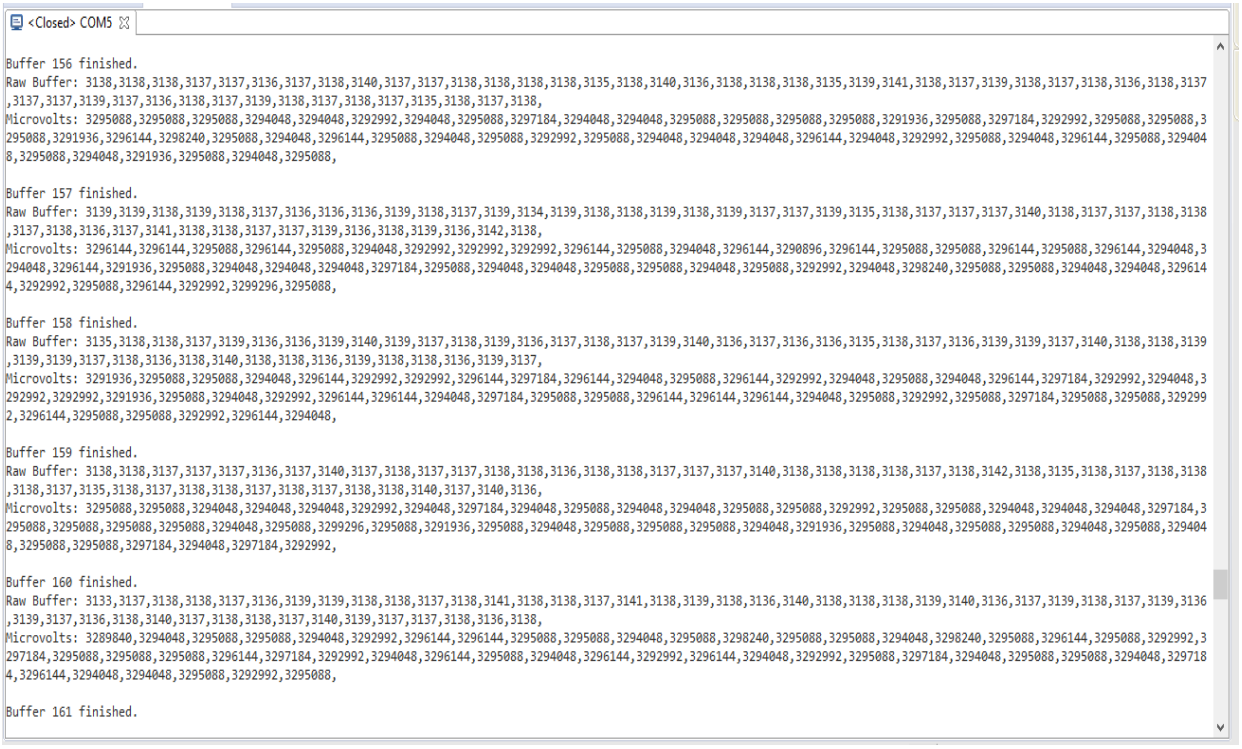

Figura 38 Resultado do ADCbufferContinuos pelo Code Composer Studio

Para uma melhor compreensão do software podemos realçar os seguintes tópicos:

- **Inicialização Pre-RTOS ;**
- **Tarefa Simple_Serial_Sockt_Server**: a tarefa com a menor prioridade, a principal da aplicação;
- **ICALL**: modulo interface responsável pela comunicação entre *Stack* de comunicação BLE e as tarefas.

A função principal (main() Figura 39) é o ponto de partida para a execução da aplicação. Neste ponto são realizadas as inicializações dos drivers e desabilitadas as interrupções da placa. A função main() é também responsável pela inicialização da gestão de energia e pela construção das tarefas. Na etapa final desta função, as interrupções são ativadas e o *scheduler* do *kernel* do SYS/BIOS é iniciado, através da invocação BIOS- start().

```
int main()
{
  /* Register Application callback to trap asserts raised in the Stack */
  RegisterAssertCback(AssertHandler);

  PIN_init(BoardGpioInitTable);

#ifdef CC1350_LAUNCHXL
  // Enable 2.4GHz Radio
  radCtrlHandle = PIN_open(&radCtrlState, radCtrlCfg);

#ifdef POWER_SAVING
  Power_registerNotify(&rFSwitchPowerNotifyObj,
                       PowerCC26XX_ENTERING_STANDBY | PowerCC26XX_AWAKE_STANDBY,
                       (Power_NotifyFxn) rFSwitchNotifyCb, NULL);
#endif //POWER_SAVING
#endif //CC1350_LAUNCHXL

#ifdef CACHE_AS_RAM
  // retain cache during standby
  Power_setConstraint(PowerCC26XX_SB_VIMS_CACHE_RETAIN);
  Power_setConstraint(PowerCC26XX_NEED_FLASH_IN_IDLE);
#else
  // Enable iCache prefetching
  VIMSConfigure(VIMS_BASE, TRUE, TRUE);
  // Enable cache
  VIMSModeSet(VIMS_BASE, VIMS_MODE_ENABLED);
#endif //CACHE_AS_RAM

#ifdef ICALL_JT
  /* Update User Configuration of the stack */
  user0Cfg.appServiceInfo->timerTickPeriod = Clock_tickPeriod;
  user0Cfg.appServiceInfo->timerMaxMillisecond  = ICall_getMaxMSecs();
#endif  /* ICALL_JT */
  /* Initialize ICall module */
  ICall_init();

  /* Start tasks of external images - Priority 5 */
  ICall_createRemoteTasks();

  /* Kick off profile - Priority 3 */
  GAPRole_createTask();

  SimpleSerialSocketServer_createTask();

  /* enable interrupts and start SYS/BIOS */
  BIOS_start();

  return 0;
}
```

Figura 39 main da aplicação

A primeira coisa que acontece é que a main.c inicializa as tarefas de suporte para a inicialização da pilha(stack) e do serviço GAP, chama o SimpleSerialSocketSer- ver_createTask() para criar a tarefa do utilizador e inicia o TI-RTOS *Scheduler*. A partir daí, SimpleSerialSocketServer_createTask() configura a tarefa do utilizador com o TI- RTOS e informa o sistema que a thread da tarefa é a função chamada SimpleSerialSocket- Server_taskFxn(). Quando BIOSstart() for chamada na main(), esta função é executada. Na função SimpleSerialSocketServer_taskFxn() é chamada a função SimpleSerialSocket_Server_Init(), que configura o *Stack* BLE, os serviços e os periféricos do hardware usados pela aplicação. Aqui também são feitos os registos dos retornos de chamadas para os eventos do sistema e, a partir daí, tudo é baseado em eventos. O SimpleSerialSocket_Server_taskFxn(), em seguida, entra num *loop* infinito, aguardando que as mensagens sejam processadas. Durante o tempo de vida da aplicação, os *callbacks* serão invocados e enviarão mensagens para o thread de tarefas para manipulação no *loop* infinito encontrado em SimpleSerialSocketServer_taskFxn(). Isso inclui trocas de dados entre o servidor e o cliente (envio e receção de dados).

O serviço fornece as seguintes funções para o envio e processamentos de fluxos de dados:

- Simple_Stream_Server_sendData (uint16_t * data, uint16_t len);
- SimpleStreamServer_processStream ();
- SimpleStreamServer_disconnectStream ().

O *software* mantém uma lista FIFO interna para dados enviados para o serviço usando SimpleStreamServer_SendData(). Quando novos dados forem enviados ao serviço, a *queue* interna será processada e o máximo de dados possível é enviado ao servidor usando notificações (isso pressupõe que o cliente ativou as notificações). Este serviço depende da memória dinâmica alocada e usará o *heap ICall* da aplicação para armazenar os da- dos alocados em *queues*. Se o serviço não puder enviar todos os dados dentro do evento de conexão atual, SimpleStream_ServerprocessStream() poderá ser chamado no espaço da aplicação para processar a *queue* interna novamente noutro momento. Se o servidor se desconectar do cliente, a *queue* interna poderá ser limpa chamando SimpleStreamServer_disconnectStream(). Para receber dados do cliente, é necessária a existência de *callbacks* a serem invocadas aquando da receção, sendo para isso é necessário registá-los após a adição do serviço. Isso é feito chamando SimpleStreamServer_RegisterAppCBs (SimpleStreamServerCBs_t * appCallbacks).

4.2.2 Comandos

Para uma comunicação efetiva foram criados um conjunto de comandos que controlam a comunicação de dados: Stop and Start, leitura e escrita de entradas e saídas digitais.

O comando "Stop", Figura 40 , tal como o próprio nome diz, tem como principal objetivo interromper o envio de dados do servidor para o cliente.

```
static void SimpleStreamServer_incomingDataCB(uint16_t connHandle,
                                               uint8_t paramID, uint16_t len,
                                               uint8_t *data)
{

          if(*data == '0'){
             ADCBuf_convertCancel(adcBuf);
          }

          if(*data == '1'){
             ADCBuf_convert(adcBuf, &continuousConversion, 1);
           }

          if(*data == '3'){
              // Indicate that the stream is enabled
              PIN_setOutputValue(ledPinHandle, Board_GLED, 1);
          }

          if(*data == '4'){
              // Indicate that the stream is enabled
              PIN_setOutputValue(ledPinHandle, Board_GLED, 0);
          }


}
```

Figura 40 Implementação dos comandos

- ***Stop and Start***: Estes comandos são responsáveis pela comunicação, isto é, pelo início e final da mesma. Estes comandos definem o envio e receção de dados e o início de conversões ADC;

  Start - 0x0001 (hexadecimal)

  Stop - 0x0000 (hexadecimal)

- **Leitura e escrita de entradas e saídas digitais:** Comando enviado pelo utilizador responsável pelo início de um treino. As tramas passam a ser enviadas do servidor para o cliente com o *byte* sinalizador a apresentar o valor 1, que corresponde ao momento em que o *led* foi ligado.

  Acionar - 0x0011 (hexadecimal)

4.2.3 Trama

Por outro lado, a produção de tramas é uma consequência da aplicação do Bluetooth. Esta é a forma como os dados são transmitidos do servidor para o cliente numa comunicação.

Figura 41 Formato de trama

Como mostra a Figura 41, a trama de dados é composta por três partes distintas: Timestamp, Sinalizador Led e Amostras.

Em primeiro lugar, a Timestamp é a componente que ocupa os primeiros 2 bytes da trama de dados. Em segundo lugar é possível encontrar o Sinalizador de Led que ocupa o terceiro byte. Os últimos 16 bytes são ocupados pelas amostras, sendo que cada uma das amostras ocupam 2 bytes.

4.2.4 Receção e Processamento – Dispositivo Móvel

Para o desenvolvimento desta aplicação Android foi necessário desenvolver um conjunto de atividades. As atividades desenvolvidas têm como função ativar a função Bluetooth no dispositivo móvel, promover a procura de dispositivos que sejam passíveis de conexão (*Scaning* e *conection*), isto é, dispositivos compatíveis e a ativar notificação aquando da receção de dados provenientes do servidor (GATT Server).

Para que seja possível fazer ativação das funcionalidades Bluetooth no dispositivo móvel, foram usadas um conjunto de permissões ao nível da aplicação. Estas permissões são a chave para que o acesso às funcionalidades Bluetooth e, por consequência não seria possível estabelecer qualquer conexão, envio e receção de dados entre o servidor e o cliente, não sendo também possível encontrar qualquer dispositivo. As permissões utilizadas nesta aplicação encontram-se representadas na Figura 42.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Figura 42 Permissões Bluetooth

A Figura 43 representa o cancelamento, ou ativação das funcionalidades Bluetooth no smartphone assim como a permissão do acesso a localização do dispositivo.

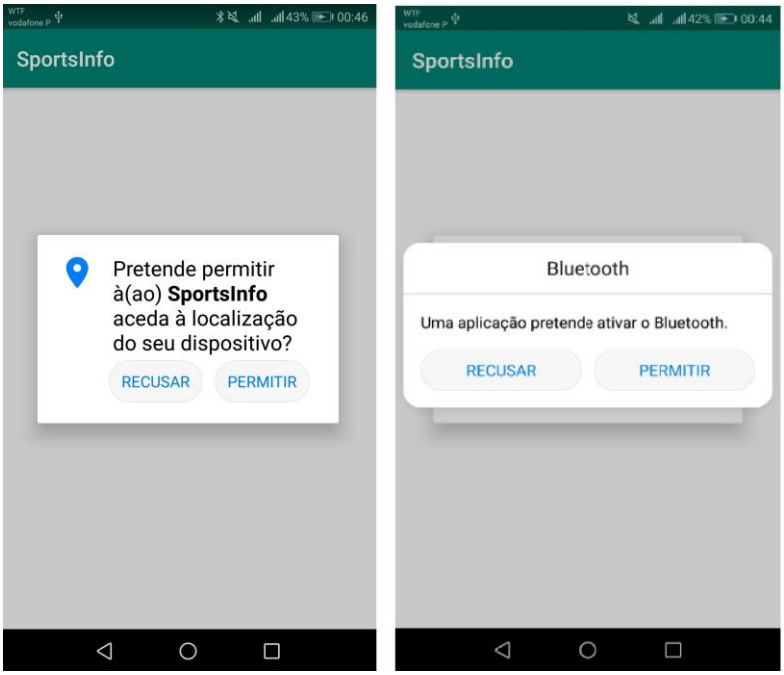

Figura 43 Ativação do Bluetooth no smartphone e acesso a localização

A pesquisa de dispositivos é uma das primeiras atividades a executar após a ativação do Bluetooth no dispositivo móvel. Essa pesquisa é realizada através do método "scanForPeripheralsWithService(UUID serviceUUIDs)" como se encontra apresentado na Figura 44 . Este método realiza a procura através do UUID dos serviços que procura. A partir deste momento a aplicação dá oportunidade ao utilizador de ativar a conexão Bluetooth ou, por outro lado, cancelar as operações e sair da aplicação, sem que haja envio ou receção de dados.

```
private BluetoothHandler(Context context) {
    this.context = context;

    // Create BluetoothCentral
    central = new BluetoothCentral(context, bluetoothCentralCallback, new Handler());

    // Scan for peripherals with a certain service UUIDs
    central.startPairingPopupHack();
    central.scanForPeripheralsWithServices(new UUID[]{SSS_SERVICE_UUID});
}
```

Figura 44 Exemplo do código da atividade Scaning

Para que a busca não seja exaustiva optou-se por fazer uma pesquisa mais focada, mais direta. Nesta aplicação, a procura é realizada apenas pelo serviço presente no sistema de aquisição de sinal (f000c0c2-0451-4000-b000-000000000000).

Depois de efetuar o *Scaning* é efetuada a conexão como se encontra apresentado na Figura 45. Depois da conexão entre o servidor e o cliente é feita a ativação da notificação e, posteriormente a receção dos dados.

```
// Callback for central
private final BluetoothCentralCallback bluetoothCentralCallback = new BluetoothCentralCallback() {

    @Override
    public void onConnectedPeripheral(BluetoothPeripheral peripheral) {
        Timber.i( message: "connected to '%s'", peripheral.getName());
    }

    @Override
    public void onConnectionFailed(BluetoothPeripheral peripheral, final int status) {
        Timber.e( message: "connection '%s' failed with status %d", peripheral.getName(), status);
    }

    @Override
    public void onDisconnectedPeripheral(final BluetoothPeripheral peripheral, final int status) {
        Timber.i( message: "disconnected '%s' with status %d", peripheral.getName(), status);

        // Reconnect to this device when it becomes available again
        handler.postDelayed(() -> {
                central.autoConnectPeripheral(peripheral, peripheralCallback);
        }, delayMillis: 5000);
    }

    @Override
    public void onDiscoveredPeripheral(BluetoothPeripheral peripheral, ScanResult scanResult) {
        Timber.i( message: "Found peripheral '%s'", peripheral.getName());
        central.stopScan();
        central.connectPeripheral(peripheral, peripheralCallback);
    }
};
```

Figura 45 Exemplo do código da atividade conexão

A aplicação BLETerm foi uma das ferramentas que auxiliaram o desenvolvimento do Software (servidor). Esta aplicação móvel foi importante no teste de verificação de envio de dados desempenhando o papel de cliente.

Capítulo 5 – Testes e Resultados

5.1 Resultados

Nesta secção são apresentados os resultados de todas as experiências realizadas durante esta dissertação. Os resultados apresentados são referentes aos comandos e à taxa de transmissão. Após a apresentação dos resultados seguem-se as conclusões.

5.1.1 Testes às taxas de transmissão

Este código foi sofrendo alterações de diversos parâmetros, entre eles, o número de amostras (bytes de dados). Após as alterações estarem efetivadas ao nível do código, era necessário recorrer a dispositivos móveis (telemóveis), que com o auxílio da aplicação BLETerm, já referida neste documento, passava a receber os dados, a partir da placa CC2640R2F, onde era possível verificar as perdas, ou não, de dados. As alterações relativas ao número de amostra fez com estas variassem entre 80 bytes, valor máximo e, 10 bytes, valor mínimo. Por outro lado, os dispositivos usados possuíam uma versão do BLE de 4.1, Huawei P8 Lite 2017, e 4.2, Huawei Mate 20 Lite. A grande diferença experimentada entre os dois dispositivos foi que o primeiro dispositivo, BLE 4.1, denotava perda de dados a uma frequência de amostragem acima de 1000 Hz. Enquanto que, o segundo dispositivo, BLE 4.2, não denotava perdas significativas de dados em frequências de amostragem até 1200 Hz usando um canal.

Os principais resultados obtidos neste teste foram de 1200 Hz, sem qualquer perda de dados e, de 10 Hz, como valor mínimo obtido. Quando se apresentam os valores de 1000 ou 1200 Hz não se está a fazer referência direta aos valores da taxa de transmissão Bluetooth. Estes valores dizem respeito à frequência com que o driver ADC converte as amostras definidas. Estas amostras são depois enviadas para uma *queue* RTOS que será processada pelo Bluetooth. Sendo assim, os valores influenciam de forma indireta os valores da taxa de transmissão BLE.

5.1.2 Testes ao Envio e Receção

Para este teste foi usado um gerador de funções a gerar uma onda quadrado com uma frequência de 50 Hz, um offset de 0 (zero) volts e 3 volts pico a pico de amplitude. No teste feito como pode ser visto na Figura 46 são apresentados os valores da onda quadrada em micro-volts, valores compreendidos aproximadamente entre 3000000 e 5000.

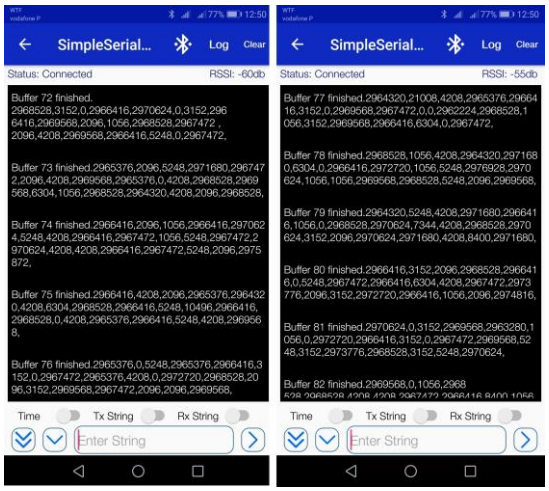

Figura 46 Resultado de teste a uma onda quadrada

5.1.3 Testes aos Comandos

Esta pequena secção diz respeitos aos testes efetuados aos comandos. Os comandos testados foram os comandos enviados do cliente para o servidor: "Stop", "Start" e "Acionar led". Pela Figura 47 é possível perceber que o comando funciona de forma correta.

Figura 47 Comando Stop

O envio de dados é imediatamente iniciado depois do utilizador escolher uma das funcionalidades quando a conexão for estabelecida, entre o cliente e o servidor. O comando "Start" pode ser utilizado após ser executado o comando Stop para voltar a enviar os dados.

Os testes efetuados a este comando mostraram também que, este comando funciona corretamente, sem que se possam aplicar falhas que causem a sua disfuncionalidade.

Por último, foi testado o comando "Acionar led" responsável por ativar um led. A ativação deste led indica o momento exato em que o utilizador pode iniciar o seu treino de reação.

Os resultados dos testes realizados ao comando "Acionar led" foram bem-sucedidos, sem que se possam apontar problemas que causem a indisponibilidade do treino de reação.

5.1.4 Evolução da perda de dados nos canais de amostragem

O teste de perdas de dados foi desenvolvido com o driver ADCBuf a executar 20 amostras, um *payload* 48 bytes e intervalo de conexão mínimo de 100 ms e máximo de 1000 ms. Como já referido, este teste foi realizado através de uma conexão estabelecida entre o servidor (sistema de aquisição de sinal) e o cliente (dispositivo móvel, o *smartphone* Huawei P20 Mate Lite). O método de transferência de dados escolhido foi a notificação já que mostra ser um método mais ágil para estabelecer a comunicação pois o servidor pode realizar o envio de dados sem o que o cliente tenha de confirmar a conexão, isto é, o cliente não envia mensagem de *acknowledgement*.

Na Figura 48 encontram-se representados os resultados ao teste de perda de dados nos canais de amostragem. Como pode ser visto, à medida que o número de canais de amostragem aumenta (aumento da frequência), a perda de dados é cada vez maior, isto é, a percentagem de perda é superior. Este fenómeno verifica-se por uma limitação da placa CC2640R2F.

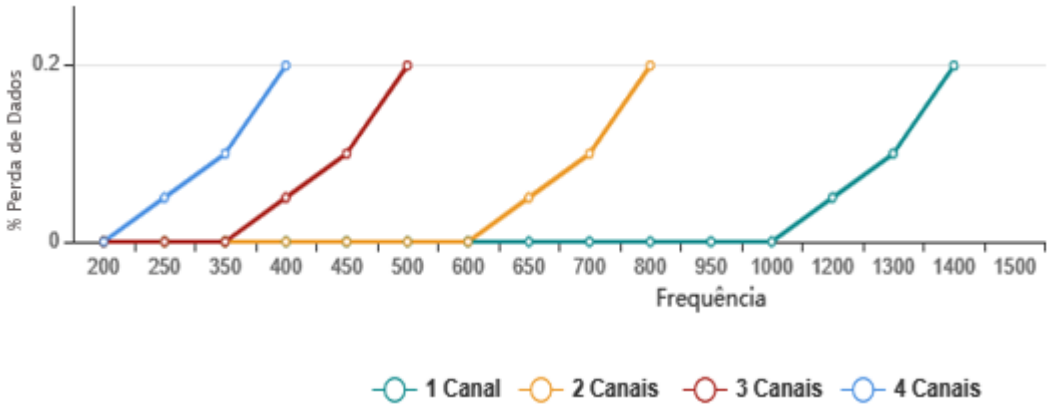

Figura 48 Evolução da perda de dados nos canais de amostragem

Como forma de conclusão desta secção, pode dizer-se que os teste desenvolvidos obtiveram na sua generalidade resultados satisfatórios. Apesar de tudo e para conceber uma abordagem mais prática e real ao teste do sistema desenvolvido houve a tentativa de introduzir testes recorrendo ao sensor. No entanto, pela falta de disponibilidade do mesmo não foi possível a sua realização, e por isso não houve a necessidade do teste ao circuito de acondicionamento, recorrendo para o tal teste a um gerador de funções.

5.1.5 Aplicação Android

Esta secção refere-se ao código implementado no capítulo 4, concretamente na secção 4.2.4 Receção e Processamento-Telemóvel. É importante referir que o papel da aplicação Android encontra-se apenas relacionado com a necessidade de teste com dispositivos Bluetooth, o que significa que o seu desenvolvimento não foi considerado prioritário para esta dissertação, já que se encontrava aliada apenas à fase de testes e, para além disso, existia a possibilidade de testar a funcionalidade do sistema desenvolvido, através da aplicação BLETerm, também esta uma aplicação Android. As principais funcionalidades desta aplicação são: estabelecer a conexão com o sistema de aquisição e condicionamento do sinal, por meio do Bluetooth Low Energy, ativando notificações que seriam desencadeadas para a receção dos dados provenientes desse mesmo sistema de aquisição, servidor na comunicação. Esta aplicação não teve o melhor dos desenvolvimentos já que havia uma falta de conhecimento e ambientação com a linguagem Android, houve uma grande dificuldade em desenvolver a conexão Android com o sistema de aquisição a partir do Bluetooth Low Energy e o tempo que se foi demonstrando curto para o desenvolvido de uma aplicação, que seria idealmente completa e bem estruturada. Mas, porém, foram implementadas algumas das funcionalidades, nomeadamente a funcionalidade que permite estabelecer conexão e ativar a notificação e receção dos dados. Ficou em falta o processa- mento dos dados, uma funcionalidade que permitirá ver por exemplo a variação dos sinais num gráfico e calcular o tempo de reação.

Capítulo 6 – Conclusão

Como forma de concluir e de refletir acerca do que foi proposto para esta dissertação apresentam-se de seguida um conjunto de secções que visam destacar os contributos, limitações, trabalho futuro e as considerações finais.

Neste projeto, a questão que se coloca relaciona-se com a implementação e utilização de um sistema de aquisição de sinal no âmbito da prática desportiva usando como mecanismo de comunicação o BLE. Como já foi referido, esta dissertação tinha como principal objetivo o desenvolvimento de um sistema de aquisição de sinal sem fios capaz de auxiliar atletas em ações desportivas. Foi conseguido um sistema de aquisição de sinal BLE com uma performance de cerca de 1000 amostras/s a 12 bits de resolução.

É possível concluir que esta dissertação cria as fundações quer do lado do cliente, quer do lado do servidor, sendo possível a criação de um sistema de aquisição de sinal com mecanismo de comunicação BLE.

6.1 Limitações

Ao longo da investigação foram sendo encontrados fatores que limitaram o desenvolvimento desta dissertação e, ao mesmo tempo, dificultaram a concretização dos objetivos da mesma dos quais se destacam a falta de conhecimento da linguagem Android e as dificuldades relacionadas com o hardware. Em primeiro lugar, a exigência temporal foi um dos fatores que dificultou o desenvolvimento deste trabalho, a falta de conhecimento na linguagem Android e a falta de prática com a ferramenta Android Studio tornaram mais difícil a concretização da aplicação móvel, a qual é um complemento importante para a compreensão dos resultados conseguidos no âmbito desta dissertação. Por outro lado, a reduzida capacidade da RAM da placa CC2640R2F não permitiu atingir uma quantidade amostras superior e frequências mais altas que permitiriam obter melhores resultados, o que poderia ajudar a retirar conclusões mais substanciais.

6.2 Trabalhos Futuros

Apesar de os objetivos para deste projeto terem sido atingidos na sua generalidade, a verdade é que esta temática e este trabalho, em específico, poderia ser completado.

Uma abordagem que poderia ser tida em conta, seria o desenvolvimento de um sistema de aquisição de sinal que faça uso da versão Bluetooth mais atual do mercado, isto é, Bluetooth 5, já que a versão utilizada nesta dissertação é a versão do Bluetooth 4.2, a qual poderá ser menos eficiente e possuir mais problemas na sua generalidade que podem já estar resolvidos na versão mais recente.

Por outro lado, a melhoria da aplicação android que poderia trazer uma maior capacidade de visualização.

6.3 Considerações Finais

Este projeto foi positivo e trouxe contributos relevantes para o autor, alargando os seus horizontes ao nível da programação quer de sistemas embebidos, quer de sistemas Android, e das tecnologias e ferramentas que o suportam. Do lado dos sistemas operativos destaca-se os conceitos de RTOS, programação de sistemas complexos de aquisição através de DMA, programação de comunicação BLE e, finalmente, o próprio ambiente de programação, o Code Composer Studio. Do lado do Android há a destacar não só o paradigma de programação bem como a ferramenta Android Studio. A interação com novas tecnologias foi o principal destaque desta dissertação, tendo a utilização dos ambientes de desenvolvimento Code Composer Studio e Android Studio foi complexa e de difícil adaptação. No entanto, permitiu desenvolver um raciocínio lógico de complexidade superior que trouxe um conhecimento de valor acrescentado. Assim sendo, conclui-se esta dissertação com a sensação de dever cumprido e com a certeza de que esta temática será o futuro dos sistemas de aquisição de sinal e de monitorização de grande parte das áreas que envolvem tecnologia.

Este documento permite perceber melhor aquela que é a aplicação de um sistema de aquisição de sinal sem fios baseado no protocolo BLE na prática desportiva.

No contexto deste projeto foi necessário desenvolver diversas competências técnicas que se vieram a mostrar valorizadoras. O desenvolvimento de conhecimentos em torno do RTOS, o contacto com novos softwares, como o Android Studio e o Code Composer Studio foram também mais valias para o desenvolvimento desta dissertação. Numa fase inicial, foram estudados os conceitos base englobados neste projeto, como RTOS e BLE, os quais tiveram um papel importante na fase de implementação e que eram desconhecidos até à realização deste projeto. Nesta fase o objetivo foi conseguir aplicar os conceitos, anteriormente estudados, num contexto aplicacional por meio da introdução do Bluetooth Low Energy e da placa CC2640R2F como a parte mais técnica e de maior complexidade do projeto, e do Android, como forma de fornecer maior funcionalidade e de conseguir obter uma interface mais *user-frendly*.

Referências

[1] Karen Lightman, "Next-Gen Sensors Make Golf Clubs, Tennis Rackets, and Baseball Bats Smarter Than Ever - IEEE Spectrum," 2016. [On- line]. Available: https://spectrum.ieee.org/consumer-electronics/gadgets/nextgen- sensors-make-golf-clubs-tennis-rackets-and-baseball-bats-smarter-than-ever

[2] Chris Smith, "Rio Olympics 2016: How wearable tech will power Team US to golds galore," 2016. [Online]. Available: https://www.wareable.com/sport/wearable-tech- at-rio-olympics-2016-2097

[3] Anonymous, "Impact Wrap Makes Your Punching Bag Smart | Technology News." [Online]. Available: https://gadgets.ndtv.com/others/news/impact-wrap- makes-your-punching-bag-smart-716060

[4] C. Lung, S. Oniga, A. Buchman, and A. Tisan, "Wireless data acquisition system for IoT applications," *Carpathian Journal of Electronic and Computer Engineering*, vol. 61, pp. 64–67, 2013. [Online]. Available: http://cjece.ubm.ro

[5] Bachuwar V.D, Shaligram A.D, and Deshmukh L P, "Low Cost Wireless Data Acquisition System for Multisensor Applications," *International Journal of Development Research*, vol. 07, no. 08, pp. 14 346–14 349, 2017. [Online]. Available: http://www.journalijdr.com

[6] W.Bluetooth, "Bluetooth Basics," *Embedded Systems programming*, pp. 9– 11, 2000. [Online]. Available: https://www.ele.uri.edu/courses/bme362/handouts/ Bluetooth.pdf

[7] Bijal Parikh, "What is Bluetooth Protocol | Basics and Working of BluetoothProtocol Explained." [Online]. Available: https://www.engineersgarage.com/ articles/bluetooth-protocol-basics-working

[8] H. Labiod, H. Afifi, and C. D. Santis, *Wi-fi , Bluetooth, Zigbee and Wimax*. Springer, 2007.

[9] K. Townsend, "Introduction to Bluetooth Low Energy,", pp. 1–11, 2014. [Online]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low- energy

[10] Anonymous, "Bluetooth® low energy physical layer," 0AD. [Online]. Available: https://microchipdeveloper.com/wireless:ble-phy-layer

[11] Texas Instruments, "CC2640/CC2650 Bluetooth low energy Software Developer's Guide (Rev. E)," Tech. Rep., 2018. [Online]. Available: http://www.ti.com/lit/ug/ swru393e/swru393e.pdf?{&}ts=1589253518308

[12] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-

power wireless technology," *Sensors*, vol. 12, no. 9, p. 11734–11753, 2012.

[13] B. Performance, W. Radio, F. Regulations, and D. Environment, "CC2640R2F SimpleLink ™ Bluetooth ® low energy Wireless MCU," *SWRS204A*, 2017. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2640r2f.pdf

[14] H. Carvalho, Y. Yao, and L. M. Gonçalves, "Flexible force sensors for e-textiles," *IOP Conference Series: Materials Science and Engineering*, vol. 254, no. 7, 2017.

[15] Texas Instruments, "TI-RTOS Overview — Bluetooth Low Energy Software Developer's Guide 3.00.00 documentation," 2016. [Online]. Available: http://software-dl.ti.com/lprf/simplelink_cc2640r2_sdk/1.00.00.22/exports/docs/blestack/html/tirtos/rtos-overview.html?fbclid=IwAR19F_QplcRXgxN1LsUeIHbgFbddjxf5DbUP55svbTOLIcg4M7Q1ymi0OEA